IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial No.:	10/719,981	Examiner	Murdough, Joshua
Applicant:	Hug, Joshua, et al.	Group Art Unit:	3621
Filed:	November 21, 2003	Atty. Docket No.:	108417.00081
Title:	Digital Rights Management for Content Rendering on Playback Devices		

DECLARATION UNDER 37 CFR 1.131

I, the undersigned, Bradley D. Hefta-Gaub, as a joint inventor of the invention claimed in application 10/719,981 (the "subject application" filed on November 21, 2003), and all claims contained therein, do hereby declare that:

1. Being hereby warned that willful false statements are punishable by fine or imprisonment, or both, under 18 U.S.C. 1001 and that such willful false statements may jeopardize the validity of the application or any resulting Letters Patent issuing thereon, I state that all statements made of my own knowledge are true and all statements made on information and belief are believed to be true.

2. The subject application was filed on November 21, 2003.

3. Development, conception, and reduction to practice of the invention claimed in the subject application, and the activities described in this declaration, took place in the Unites States of America.

4. The subject application, as filed, included thirty-one claims.

5. Of the thirty-one pending claims, nine claims (i.e., claims 1, 15, 22, 26, 27, 28, 29, 30, and 31) are independent claims and twenty-two claims (i.e., claims 2-14, 16-21 and 23-25) are dependant claims.

6. CONCEPTION:

7. The subject matter claimed in the subject application was conceived prior to July 5, 2003.

8. Prior to July 5, 2003, the invention claimed in the subject application was disclosed in an Invention Disclosure Form. A true and accurate copy of the Invention Disclosure Form is attached as Exhibit A, except that the copy has been redacted to remove all references to dates. The Invention Disclosure Form (i.e., Exhibit A) evidences the conception of the invention claimed in the subject application, which occurred prior to July 5, 2003.

9. Claim 1 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

1. In a client device, a method comprising[.] (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Obtain a new device"; "Register device with a DMS"; "play the media consuming the intrinsic rights the device has").

receiving a request for playback of digital audio or video content stored on the device; (See *Id.*: "Play the media consuming the intrinsic rights the device has")

determining an allotted playback duration for the device[.] (See *Id.*: "The most common and maybe the only set of rights that a device will acknowledge will be playback time . . .Devices will be created with a pre-determined number of allowed playback hours")

determining an elapsed playback duration for the device, the elapsed playback duration representing an amount of time previously consumed by the device while rendering digital audio or video content[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged").

determining whether a predetermined relationship between the elapsed playback duration and the allotted playback duration for the device is satisfied[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their

rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged").

regulating playback of at least the requested digital audio or video content if the predetermined relationship between the elapsed playback duration and the allotted playback duration for the device is determined to be satisfied. (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

10. Claim 2 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

2. The method of claim 1, wherein the request for playback of digital audio or video content is received via a user input device. (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Obtain a new device"; "Register device with a DMS"; "play the media consuming the intrinsic rights the device has"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

11. Claim 4 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

4. The method of claim 1, wherein playback of the requested digital audio or video content track is denied if it is determined that the relationship between the allotted playback duration and elapsed playback duration is satisfied. (See Exhibit A § 1: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed

playback hours. After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

12. Claim 5 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

> 5. The method of claim 4, further comprising: facilitating playback of the digital audio content if it is determined that the elapsed playback duration does not exceed the allotted playback duration. (See Exhibit A § 1: "Play the media consuming the intrinsic rights the device has").

13. Claim 6 was conceived prior to July 5, 2003. Claim 6 recites:

> 6. The method of claim 4, further comprising: indicating to the user at least one of the elapsed playback duration and the allotted playback duration.

Although Exhibit A does not explicitly disclose indicating playback durations to a user, the concept of indicating playback durations to a user was contemplated and conceived by the inventors prior to July 5, 2003.

14. Claim 7 was conceived prior to July 5, 2003. Claim 7 recites:

> 7. The method of claim 4, further comprising: indicating to the user the elapsed playback duration in relation to the allotted playback duration.

Although Exhibit A does not explicitly disclose indicating to the user the elapsed playback duration in relation to the allotted playback duration, the concept of indicating to the user the elapsed playback duration in relation to the allotted playback duration was contemplated and conceived by the inventors prior to July 5, 2003.

15. Claim 8 was conceived prior to July 5, 2003. Claim 8 recites:

8. The method of claim 7, wherein the digital audio or video content is encoded in accordance with at least one of an advanced audio encoding algorithm, and adaptive multi-rate encoding algorithm, and an MP3 encoding algorithm.

Although Exhibit A does not explicitly disclose that the digital audio or video content is encoded, the concept of the audio of video content being encoded was contemplated and conceived as part of the invention by the inventors prior to July 5, 2003.

16. Claim 9 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

9. The method of claim1, further comprising: denying playback of the requested digital audio or video content if the elapsed playback duration added to a run length associated with the requested content exceeds the allotted playback duration. (See Exhibit A § 1: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

17. Claim 10 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

10. The method of claim 1, further comprising: denying playback of additional digital audio or video content stored on the device in addition to the requested digital audio or video content if it is determined that the elapsed playback duration is equal to or exceeds the allotted playback duration. (See Exhibit A § 1: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-

charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

18. Claim 11 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

    11. The method of claim 1, wherein the allotted playback duration is determined based upon predetermined rights associated with the device. (See Exhibit A § 1: "Devices will be created with a pre-determined number of allowed playback hours").

19. Claim 12 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

    12. The method of claim 1, wherein the allotted playback duration is determined based upon data received from the content rights server. (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "At some point either automatically or at the users request, the device will be issued a refresh token"; "If the user refuses to refresh/recharge their device, does not connect their device to a DMS, cancels their subscription to their DMS . . . the device will refuse to playback the user's subscription content").

20. Claim 13 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

    13. The method of claim 1, further comprising: periodically increasing the allotted playback duration prior to the allotted playback duration exceeding the elapsed playback duration. (See Exhibit A § 1: "At some point either automatically or at the users request, the device will be issued a refresh token").

21. Claim 14 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

> 14. The method of claim 10, wherein the allotted playback duration is increased based upon entitlements granted to the user by a service provider. (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "At some point either automatically or at the users request, the device will be issued a refresh token"; "If the user refuses to refresh/recharge their device, does not connect their device to a DMS, cancels their subscription to their DMS . . . the device will refuse to playback the user's subscription content").

22. Claim 15 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

> 15. In a digital content rendering device, a method comprising[.] (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Obtain a new device"; "Register device with a DMS"; "play the media consuming the intrinsic rights the device has").
>
> rendering one of a plurality of audio or video content items[.] (See *Id.*: "Play the media consuming the intrinsic rights the device has")
>
> determining an elapsed playback duration for which digital audio or video content has been rendered[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged").
>
> regulating further content rendering by the digital content rendering device if the elapsed playback duration satisfies a predetermined relationship with respect to an allotted playback duration. (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-

determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

23. Claim 16 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

    16. The method of claim 15, wherein the elapsed playback duration represents an amount of time for which content has been rendered by the digital content rendering device. (See Exhibit A § 1: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged")

24. Claim 18 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

    18. The method of claim 15, wherein regulating comprises denying further content rendering by the digital content rendering device if the elapsed playback duration satisfies a predetermined relationship with respect to the allotted playback duration. (See Exhibit A § 1: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

25. Claim 19 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

19. The method of claim 15, wherein the allotted playback duration represents at least one of an amount of render time for which content may be rendered on the digital content rendering device, and a quantity of data that may be processed by the digital content rendering device to render content on the device. (See Exhibit A § 1: "The most common and maybe the only set of rights that a device will acknowledge will be playback time . . .Devices will be created with a pre-determined number of allowed playback hours")

26. Claim 20 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

20. The method of claim 19, further comprising: facilitating playback of the digital audio content if it is determined that the elapsed playback duration does not exceed the amount of render time corresponding to allotted playback right. (See Exhibit A § 1: "Play the media consuming the intrinsic rights the device has").

27. Claim 22 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

22. In a digital content rendering device, a method comprising[.] (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Obtain a new device"; "Register device with a DMS"; "play the media consuming the intrinsic rights the device has").

identifying a playback right associated with the digital content rendering device representing an allotted measure of digital audio or video content that may be rendered by the digital content rendering device[.] (See *Id.*: "The most common and maybe the only set of rights that a device will acknowledge will be playback time . . .Devices will be created with a pre-determined number of allowed playback hours").

determining whether the allotted measure of content has been rendered by the device[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged").

preventing further content rendering on the digital content rendering device if it is determined that the allotted measure of digital audio or video content that may be rendered by the digital content rendering device has previously been rendered by the device. (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

28. Claim 23 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

> 23. The method of claim 22, wherein the allotted measure of digital audio or video content that may be rendered represents an amount of time that the digital content rendering device may render the digital audio or video content. (See Exhibit A § 1: "The most common and maybe the only set of rights that a device will acknowledge will be playback time[.]" . . .Devices will be created with a pre-determined number of allowed playback hours").

29. Claim 24 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

> 24. The method of claim 22, wherein the playback right associated with the digital content rendering device is further associated with a user, and wherein the user is denied playback of any additional content items by the digital content rendering

device once it is determined that the allotted measure of digital audio or video content that may be rendered by the digital content rendering device has previously been rendered by the device. (See Exhibit A § 1: "At some point either automatically or at the users request, the device will be issued a refresh token"; "If the user refuses to refresh/recharge their device, does not connect their device to a DMS, cancels their subscription to their DMS . . . the device will refuse to playback the user's subscription content").

30. Claim 25 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

25. The method of claim 24, wherein the playback right is determined based upon a subscription agreement between the user and a content provider. (See Exhibit A § 1: "At some point either automatically or at the users request, the device will be issued a refresh token"; "If the user refuses to refresh/recharge their device, does not connect their device to a DMS, cancels their subscription to their DMS . . . the device will refuse to playback the user's subscription content").

31. Claim 26 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

26. A digital content rendering apparatus comprising[.] (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Obtain a new device"; "Register device with a DMS"; "play the media consuming the intrinsic rights the device has").

a storage medium having stored thereon programming instructions designed to enable the apparatus to[.] (See *Id.*: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Any communication medium should be usable: WiFi, WAN, GPRS, Lan, USB, 1394, SMS, MMS etc.") (Although Exhibit A does not explicitly disclose programming instructions

stored on a storage medium or executed by a processor, the inventors assert that at the time of conception, the inventors contemplated and conceived a software embodiment of the invention, which included programming instructions stored on a storage medium and executed by a processor, i.e. software stored in a memory).

receive a request for playback of digital audio or video content stored on the apparatus[.] (See *Id.*: "Play the media consuming the intrinsic rights the device has").

determine an allotted playback duration for the apparatus[.] (See *Id.*: "The most common and maybe the only set of rights that a device will acknowledge will be playback time . . .Devices will be created with a pre-determined number of allowed playback hours").

determine an elapsed playback duration for the apparatus, the elapsed playback duration representing an amount of time previously consumed by the apparatus while rendering digital audio or video content[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged").

determine whether a predetermined relationship between the elapsed playback duration and the allotted playback duration for the apparatus is satisfied[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged").

regulate playback of at least the requested digital audio or video content if the predetermined relationship between the elapsed playback duration and the allotted playback duration for the apparatus is determine to be satisfied[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours.

After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

at least one processor coupled with the storage medium to execute the programming instructions. (See *Id.*: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Any communication medium should be usable: WiFi, WAN, GPRS, Lan, USB, 1394, SMS, MMS etc.") (Although Exhibit A does not explicitly disclose programming instructions stored on a storage medium or executed by a processor, the inventors assert that at the time of conception, the inventors contemplated and conceived a software embodiment of the invention, which included programming instructions stored on a storage medium and executed by a processor, i.e. software stored in a memory).

32. Claim 27 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

27. A digital content rendering apparatus comprising[.] (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Obtain a new device"; "Register device with a DMS"; "play the media consuming the intrinsic rights the device has").

a storage medium having stored therein programming instructions designed to enable the apparatus to[.] (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Any communication medium should be usable: WiFi, WAN, GPRS, Lan, USB, 1394, SMS, MMS etc.") (Although Exhibit A does not explicitly disclose programming instructions stored on a storage medium or executed by a processor, the inventors assert that at the time of conception, the inventors contemplated and conceived a software embodiment of the invention, which included programming instructions stored on a storage medium and executed by a processor, i.e. software stored in a memory).

render one of a plurality of audio or video content items[.] (See *Id.*: "Play the media consuming the intrinsic rights the device has")

determine an elapsed playback duration for which digital audio or video content has been rendered[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged").

regulate further content rendering by the digital content rendering apparatus if the elapsed playback duration satisfies a predetermined relationship with respect to an allotted playback duration[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

at least one processor coupled with the storage medium to execute the programming instructions. (See *Id.*: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Any communication medium should be usable: WiFi, WAN, GPRS, Lan, USB, 1394, SMS, MMS etc.") (Although Exhibit A does not explicitly disclose programming instructions stored on a storage medium or executed by a processor, the inventors assert that at the time of conception, the inventors contemplated and conceived a software embodiment of the invention, which included programming instructions stored on a storage medium and executed by a processor, i.e. software stored in a memory).

33. Claim 28 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

28. A digital content rendering apparatus comprising[.] (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Obtain a new device"; "Register device with a DMS"; "play the media consuming the intrinsic rights the device has").

a storage medium having stored thereon programming instructions designed to enable the digital content rendering apparatus to[.] (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Any communication medium should be usable: WiFi, WAN, GPRS, Lan, USB, 1394, SMS, MMS etc.") (Although Exhibit A does not explicitly disclose programming instructions stored on a storage medium or executed by a processor, the inventors assert that at the time of conception, the inventors contemplated and conceived a software embodiment of the invention, which included programming instructions stored on a storage medium and executed by a processor, i.e. software stored in a memory).

identify a playback right associated with the digital content rendering apparatus representing an allotted measure of digital audio or video content that may be rendered by the digital content rendering apparatus[.]

determine whether the allotted measure of content has been rendered by the apparatus[.] (See *Id.*: "The most common and maybe the only set of rights that a device will acknowledge will be playback time . . .Devices will be created with a pre-determined number of allowed playback hours")

prevent further content rendering on the digital content rendering apparatus if it is determined that the allotted measure of digital audio or video content that may be rendered by the digital content rendering apparatus has previously been rendered by the apparatus[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to

recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

at least one processor coupled with the storage medium to execute the programming instructions. (See *Id.*: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Any communication medium should be usable: WiFi, WAN, GPRS, Lan, USB, 1394, SMS, MMS etc.") (Although Exhibit A does not explicitly disclose programming instructions stored on a storage medium or executed by a processor, the inventors assert that at the time of conception, the inventors contemplated and conceived a software embodiment of the invention, which included programming instructions stored on a storage medium and executed by a processor, i.e. software stored in a memory).

34. Claim 29 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

29. A machine readable medium having stored thereon machine executable instructions, the execution of which to implement a method comprising[.] (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device (b) digital media service (DMS)"; "Any communication medium should be usable: WiFi, WAN, GPRS, Lan, USB, 1394, SMS, MMS etc.") (Although Exhibit A does not explicitly disclose programming instructions stored on a storage medium or executed by a processor, the inventors assert that at the time of conception, the inventors contemplated and conceived a software embodiment of the invention, which included programming instructions stored on a storage medium and executed by a processor, i.e. software stored in a memory).

receiving a request for playback of digital audio or video content stored on the device[.] (See *Id.*: "Play the media consuming the intrinsic rights the device has").

determining an allotted playback duration for the device[.] (See *Id.*: "The most common and maybe the only set of rights that a device will acknowledge will be

playback time . . .Devices will be created with a pre-determined number of allowed playback hours").

determining an elapsed playback duration for the device, the elapsed playback duration representing an amount of time previously consumed by the device while rendering digital audio or video content[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged").

determining whether a predetermined relationship between the elapsed playback duration and the allotted playback duration for the device is satisfied[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged").

regulating playback of at least the requested digital audio or video content if the predetermined relationship between the elapsed playback duration and the allotted playback duration for the device is determined to be satisfied. (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

35. Claim 30 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

30. A machine readable medium having stored thereon machine executable instructions, the execution of which to implement a method comprising[.] (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device

(b) digital media service (DMS)"; "Any communication medium should be usable: WiFi, WAN, GPRS, Lan, USB, 1394, SMS, MMS etc.") (Although Exhibit A does not explicitly disclose programming instructions stored on a storage medium or executed by a processor, the inventors assert that at the time of conception, the inventors contemplated and conceived a software embodiment of the invention, which included programming instructions stored on a storage medium and executed by a processor, i.e. software stored in a memory).

rendering one of a plurality of audio or video content items[.] (See *Id.*: "Play the media consuming the intrinsic rights the device has").

determining an elapsed playback duration for which digital audio or video content has been rendered[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged").

regulating further content rendering by the digital content rendering device if the elapsed playback duration satisfies a predetermined relationship with respect to an allotted playback duration. (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

36. Claim 31 was conceived prior to July 5, 2003, as shown by Exhibit A. The following mapping of the claim elements to Exhibit A is an example of how Exhibit A shows conception of the claim:

31. A machine readable medium having stored thereon machine executable instructions, the execution of which to implement a method comprising[.] (See Exhibit A § 1: "There are two primary actors involved in this invention, (a) device

(b) digital media service (DMS)"; "Any communication medium should be usable: WiFi, WAN, GPRS, Lan, USB, 1394, SMS, MMS etc.") (Although Exhibit A does not explicitly disclose programming instructions stored on a storage medium or executed by a processor, the inventors assert that at the time of conception, the inventors contemplated and conceived a software embodiment of the invention, which included programming instructions stored on a storage medium and executed by a processor, i.e. software stored in a memory).

identifying a playback right associated with the digital content rendering device representing an allotted measure of digital audio or video content that may be rendered by the digital content rendering device[.] (See *Id.*: "The most common and maybe the only set of rights that a device will acknowledge will be playback time . . .Devices will be created with a pre-determined number of allowed playback hours").

determining whether the allotted measure of content has been rendered by the device[.] (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged").

preventing further content rendering on the digital content rendering device if it is determined that the allotted measure of digital audio or video content that may be rendered by the digital content rendering device has previously been rendered by the device. (See *Id.*: "When these rights are used up, the device will require users to recharge their rights"; "Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged"; "If a user refuses to recharge/refresh their device . . . the device will refuse to playback the user's subscription content").

37. REDUCTION TO PRACTICE

38. Diligent pursuit of actual reduction to practice of the invention claimed in the subject application began prior to July 5, 2003, and continued, without lapse, through the subject application filing date of November 21, 2003.

39. Exhibits B through I show that employees of RealNetworks, Inc. ("RealNetworks") continued to work on implementing the claimed invention and other features which, although not directly claimed in the subject application, were required in order to implement the claimed invention.

40. Exhibits B through I reference the Orange Protocol. "Orange" is a code name that was used by employees of RealNetworks to refer to the portions of the Helix Device DRM project which included work on necessary prerequisites for the invention claimed in the subject application. Other portions of the Helix Device DRM project implemented the invention claimed in the subject application after the necessary prerequisites had been implemented.

41. The Orange Protocol is a licensing protocol based on user subscriptions and allotted playback-time based on user subscriptions, as opposed to traditional content licensing models where licensing is based on media content or media files. The base components of the Orange Protocol are components that send an authenticated subscription or playback-time license to a device in a flexible and secure manner.

42. In order to implement the claimed invention, it was necessary to first implement the base components of the Orange Protocol because a) the base components facilitate secure, authenticated transmission of messages to the device, a necessary prerequisite for services that interact with devices in the embodiment of the claimed invention we were reducing to practice, and b) the base components provide the basic building blocks of a subscription or user-account subscription license, which is a necessary prerequisite to tracking allocation of elapsed playback time by the user-account in the embodiment of the claimed invention we were reducing to practice.

43. A true and accurate copy of an email dated June 20, 2003 is attached as Exhibit B. The email is from Alain Hamel, a RealNetworks employee who worked on implementing

the Orange Protocol. The email is addressed to Adam Cappio, a RealNetworks employee who worked on the Orange Protocol either directly or indirectly. The "Original Message" listed in the email is another email from Rahul Agarwal, a RealNetworks employee, to other RealNetworks employees who worked on the Orange Protocol either directly or indirectly. One of the recipients of the "Original Message" is Joshua Hug, a joint inventor of the subject application. The email includes an attachment, Orange.doc Version 0.1, which is a draft document describing some features of the Orange Protocol. The email and the attachment show that employees of RealNetworks continued to work on and revise the Orange Protocol around the time of June 20, 2003. The date printed at the bottom corner of the exhibit (5/19/2009) is the date the document was printed by the Applicants' attorney.

44. A true and accurate copy of an email dated July 1, 2003 is attached as Exhibit C. The email is from Qiang Luo, a RealNetworks employee who worked on implementing the Orange Protocol. The email is addressed to employees of RealNetworks who worked on the Orange Protocol either directly or indirectly. One of the recipients of the email is Joshua Hug, a joint inventor of the subject application. The email states "this is required for the Orange protocol." This email shows that employees of RealNetworks were communicating about and working on the Orange Protocol and the claimed invention around the time of July 1, 2003.

45. A true and accurate copy of an email dated July 16, 2003 is attached as Exhibit D. The email is from Joshua Hug, one of the inventors of the claimed invention. The email is addressed to RealNetworks employees who worked on the Orange Protocol either directly or indirectly. The email includes an attachment, Orange v.12.doc, which is a draft document describing some features of the Orange Protocol. The email and the attachment show that employees of RealNetworks continued to work on and revise the Orange Protocol and the claimed invention around the time of July 16, 2003. The date printed at the bottom corner of the exhibit (5/19/2009) is the date the document was printed by the Applicants' attorney.

46. Although I resigned from RealNetworks in August of 2003, Exhibit E is a copy of an email that RealNetworks provided to me for my review. I believe that Exhibit E is a true and accurate copy of an email dated August 1, 2003. The email is from Sheldon Fu, a RealNetworks employee who worked on the Orange Protocol. The email is addressed to employees of RealNetworks who worked on implementing the Orange Protocol either directly or indirectly. One of the recipients of the email is Joshua Hug, a joint inventor of the subject application. The email states "Josh pointed out that our current Orange protocol is vulnerable to replay attack[.]" I believe that this email shows that employees of RealNetworks were working on the Orange Protocol and the claimed invention around the time of August 1, 2003.

47. Although I resigned from RealNetworks in August of 2003, Exhibit F is a copy of an email that RealNetworks provided to me for my review. I believe that Exhibit F is a true and accurate copy of an email dated August 2, 2003. The email is from Sheldon Fu, a RealNetworks employee who worked on the Orange Protocol. The email is addressed to employees of RealNetworks who worked on implementing the Orange Protocol either directly or indirectly. One of the recipients of the email is Joshua Hug, a joint inventor of the subject application. The email includes an attachment, Orange v.14a.doc, which I believe is a draft document describing some features of the Orange Protocol. I believe that the email and the attachment show that employees of RealNetworks continued to work on and revise the Orange Protocol and the claimed invention around the time of August 2, 2003. The date printed at the bottom corner of the exhibit (5/19/2009) is the date the document was printed by the Applicants' attorney.

48. Although I resigned from RealNetworks in August of 2003, Exhibit G is a copy of an email that RealNetworks provided to me for my review. I believe that Exhibit G is a true and accurate copy of an email dated August 7, 2003. The email is from Joshua Hug, one of the inventors of the claimed invention. The email is addressed to RealNetworks employees who worked on the Orange Protocol either directly or indirectly. The email includes two attachments, helix device DRM.vsd and Helix Device DRM v.21.doc, which I believe are draft documents describing some features

of the Helix Device DRM, also known as the Orange Protocol. I believe that the email and the attachment show that employees of RealNetworks continued to work on and revise the Orange Protocol and the claimed invention around the time of August 7, 2003. The date printed at the bottom corner of the exhibit (5/19/2009) is the date the document was printed by the Applicants' attorney.

49. Although I resigned from RealNetworks in August of 2003, Exhibit H is a copy of an email that RealNetworks provided to me for my review. I believe that Exhibit H is a true and accurate copy of an email dated August 15, 2003. The email of August 15, 2003 included two PowerPoint™ file attachments, also included in Exhibit H. The email is from Joshua Hug, one of the inventors of the claimed invention. I believe that the PowerPoint attachments are presentations related to some features of the Helix Device DRM, also known as the Orange Protocol. The second attachment, titled "The Music Experience w/ Helix DRM," states:

    - Designed to allow for the dynamic update of a group of licenses on a recurring basis
    - As long as your subscription is current, you won't have to re-license your content.

I believe that the "subscription" referenced in the attachment is a discrete, rechargeable time period for which a user may play media, and is related to the allotted and elapsed playback times of the claims. For example, "Page 7" of the second attachment, titled "The Music Experience w/ Helix DRM," discusses that individual per-track music licenses are grouped into "subscriptions," that "[a]ccess to the group of licensing is provisioned and managed by a single parent "subscription" license," and states, "[a]s long as your subscription is current, you won't have to re-license your content." The next pages, "Page 8," "Page 9," and "Page 10," discuss "User Licensing" ("[s]ame benefits as Subscription licensing, plus much simpler portability and rights management"), also implemented by the Orange Protocol. I believe that the email and attachments show that employees of RealNetworks were working on some features of the Helix Device DRM, also know as the Orange Protocol, around the time of August

15, 2003. The date printed at the bottom corner of the exhibit (September 30, 2008) is the date the document was printed by the Applicants' attorney.

50. Although I resigned from RealNetworks in August of 2003, Exhibit I is a copy of an email that RealNetworks provided to me for my review. I believe that Exhibit I is a true and accurate copy of an email dated February 9, 2004. The email is from Joshua Hug, one of the inventors of the claimed invention. The email is addressed to RealNetworks employees who I believe worked on the Orange Protocol either directly or indirectly. The email includes an attachment, Helix Device DRM v.4.doc, which is a draft document describing some features of the Helix Device DRM Protocol, which features included the Orange Protocol and the claimed invention. I believe that the email and the attachment show that other employees of RealNetworks continued to work on and revise the Orange Protocol and the claimed invention around the time of February 9, 2004. I believe that the email and attachment also show that other employees of RealNetworks continued to pursue actual reduction to practice of the claimed invention after the filing date of the subject application. I believe that the email and attachment also show the link between the base components of the Orange Protocol and the claimed invention. See e.g. Exhibit I §§ 3.1, 4.3, 4.4. The date printed at the top and bottom corners of the exhibit (5/19/2009) is the date the document was printed by the Applicants' attorney.

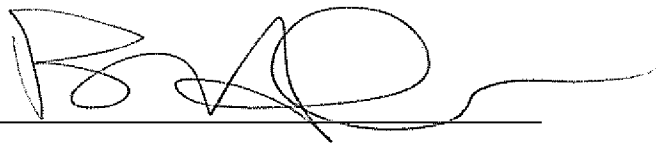
51. CONSTRUCTIVE REDUCTION TO PRACTICE

52. Constructive reduction to practice of the claimed invention occurred no later than November 21, 2003, when the subject application was filed with the United States Patent and Trademark office.

53. I believe that the Invention Disclosure Form (Exhibit A) and the emails (Exhibits B through I) also evidence the diligent pursuit of actual reduction to practice, which began prior to July 5, 2003, and continued through the November 21, 2003 filing date of the subject application.

8/3/09

Date



Bradley D. Hefta-Gaub

EXHIBIT A

REALNETWORKS INVENTION DISCLOSURE FORM

ATTORNEY-CLIENT PRIVILEGED COMMUNICATION

DATE: [redacted]

It is important to provide accurate and detailed information on this form. The information will be used to evaluate your invention for possible filing as a patent application and is necessary to help us complete the application. When completed and signed, please return this form to Lisa Benner, **Legal Department**. If you have any questions, please call Steven Stewart, x6467.

1. Inventor: Hug (Last Name) Joshua (First Name) D (Middle Initial)

Phone 206-892-6621 Location: Seattle Fax #

Citizenship: USA Contractor: YES ____ NO X

Inventor E-Mail Address: jhug@real.com

Home Address: 1101 Queen Anne Ave. APT 202

City Seattle State WA Zip 981009 Country USA

*Corporate Level Division (e.g. Media Systems) Media Systems Sub -Division Client Dev Group

Supervisor* Rahul Agarwal Phone 206-674- 2359 Location: Seattle

Inventor: Hefta-Gaub (Last Name) Bradley (First Name) D (Middle Initial)

Phone 206-674-2272 Location: Seattle Fax #

Citizenship: USA Contractor: YES ____ NO X

Inventor E-Mail Address: brad@real.com

Home Address: 3939 Eastern Ave. N.

City Seattle State WA Zip 98103 Country USA

*Corporate Level Division (e.g. Media Systems) Media Systems Sub -Division Codec

Supervisor* Martin Plaehn Phone ____ Location: Seattle

***If you are unsure of this information, please discuss with your manager.**

(PROVIDE SAME INFORMATION AS ABOVE FOR EACH ADDITIONAL INVENTOR)

2. Title of Invention: Method of expiring and renewing subscription media

3. Describe the technology/product/process (code name) the invention relates to (be specific if you can): Portable music & media subscription devices.

4. Include several key words to describe the technology area of the invention in addition to # 3 above: DRM, Rights enforcement, secure transfer, portable devices, rights provisioning

5. Stage of development (i.e. % complete, simulations done, alpha code if any, etc.): Idea has been documented in slides, no code has been created nor has this been shared with an external party.

6. (a) Has a description of your invention been, or will it shortly be, published outside of Real Networks?

NO: X YES: ____ If YES, was the manuscript submitted to Legal for pre-publication approval? ____

IDENTIFY THE PUBLICATION AND THE DATE PUBLISHED (TO BE PUBLISHED): ____

(b) Has your invention been used/sold or planned to be used/sold by Real Networks or others?

NO: X YES: ____ DATE WAS OR WILL BE SOLD: ____

(c) Does this invention relate to technology that is or will be covered by a SIG (special interest group)/standard/ or specification?

NO: _____ YES: ___X___ Name of SIG/Standard/Specification: Future Open mobile alliance DRM standards... (not currently in scope, but after we demonstrate I assume they will jump all over this.)

(d) If the invention is software, actual or anticipated date of any beta tests outside RealNetworks: Late Summer/Fall 2003

7. Was the invention conceived or constructed in collaboration with anyone other than a RealNetworks employee or in performance of a project involving entities other than RealNetworks, e.g. government, other companies, universities or consortia?

NO: ___X___ YES: _____ Name of individual or entity: _____

8. Is this invention related to any other invention disclosure that you or anyone else has recently submitted? If so, please give the title and inventors names: No.

PLEASE READ AND FOLLOW THE DIRECTIONS ON HOW TO WRITE A DESCRIPTION OF YOUR INVENTION

Attach a description of the invention to this form, DATED AND SIGNED BY AT LEAST ONE PERSON WHO IS NOT A NAMED INVENTOR, and include the following information:

1. **Describe in detail what the components of the invention are and how the invention works.**
2. **Describe advantage(s) of your invention over what is done now. (E.g. What problems does it solve, if any?)**
3. **YOU MUST include at least one figure illustrating the invention. If the invention relates to software, include a flowchart or pseudo-code representation of the algorithm.**
4. **Value of your invention to RealNetworks (how will or could it be used?).**
5. **Explain how your invention is novel. If the technology itself is not new, explain what makes it different.**
6. **Identify the closest or most pertinent prior art that you are aware of from RealNetworks or anyone else.**
7. **Who is likely to want to use this invention or infringe the patent if one is obtained and how would infringement be detected?**

***HAVE YOUR SUPERVISOR READ, DATE AND SIGN COMPLETED FORM**

DATE: _____ SUPERVISOR: _____

BY THIS SIGNING, I (SUPERVISOR) ACKNOWLEDGE THAT I HAVE READ AND UNDERSTAND THIS DISCLOSURE

1. **Describe in detail what the components of the invention are and how the invention works.**

This invention applies the concept of a rechargeable battery to Digital Media Expiration. Expiring content in an acceptable manor is a very hard problem. This invention creates a simple yet highly effective solution to this problem.

There are two primary actors involved in this invention, (a) device (b) digital media service (DMS). An optional actor (c) is a pass through intermediary device (ie a Personal computer).

Linking the actors is a communication protocol. These Actors have been linked together in prior art. This invention focuses on the goal of the communication protocol these actors communicate through. Prior art has tried to transfer actual rights information from a DMS to a device. This has proven, (a) hard to define in a secure manor, (b) hard to create devices with the required components (c) expensive (d) a complicated user experience.

This invention removes the need of a DMS to provision rights to a device. Instead, a simple recharge command is all that needs to be transmitted. Device's will be manufactured/upgraded to contain an intrinsic set of rights they know how to enforce. Those rights will never change, and will always be enforced when a device consumes content.

Our invention is to create devices with set rights already in the devices. When these rights are used up, the device will require users to recharge their rights. The DMS will simply need to generate a refresh token for the device they want to recharge.

The most common and maybe the only set of rights that a device will acknowledge will be playback time. Again this is related to the idea of using a battery for digital rights expiration. Devices will be created with a pre-determined number of allowed playback hours. After that time has passed, the devices will refuse to play content until they are re-charged. Recharging a device can be a very simple straightforward process for all actors involved.

The following flow is imagined:

- Obtain a new device
- Register device with a DMS
- DMS sends a recharge token to the device
    - Numerous protocols are possible some of which are claims of this invention
    - Any communication medium should be usable: WiFi, WAN, GPRS, Lan, USB, 1394, SMS, MMS etc.
- Freely transfer subscription media to devices.
    - No protocol is required
    - Again any communication medium, examples are: WiFi, WAN, GPRS, Lan, USB, 1394, etc.
- Play the media consuming the intrinsic rights the device has

- At some point either automatically or at the users request, the device will be issued a refresh token.
- If the user refuses to refresh/recharge their device, does not connect their device to a DMS, cancels their subscription to their DMS, hard re-sets their device, etc, The device will refuse to playback the user's subscription content. Users will understand this because the device will behave as if the battery has run out on their device.

Protocol claims:
Numerous methods can be imagined to transfer a refresh command to a device. The protocol must ensure it cannot be re-played by a man in the middle. The DMS must be able to authenticate the identity of the device. The device must authenticate a valid DMS is re-charging/refreshing their rights.
This could be accomplished via:

- A shared secrete between the DMS & device
- A bi-directional public/private key exchange between the DMS & device
- A one-way public-key/certificate exchange between the DMS and the device.
- Or a one-way public-key/certificate exchange between the device and DMS.
- A symmetric key exchange protocol between the DMS and the device.
- The protocols above but replacing the DMS with an intermediary device (ie a PC).
- Any other protocol that full-fills the requirements listed above.

2. Describe advantage(s) of your invention over what is done now. (E.g. What problems does it solve, if any?)

This invention has numerous advantages over what is done now.

This invention greatly reduces the engineering requirements and BOM required to create secure subscription devices. Thus in the end lowering the cost users will have to pay for these devices and increasing the profit margin for manufactures of subscription devices. Currently the industry is focused on building secure clocks into secure devices. Some companies have been pushing this for a LONG time but the engineering problems have been significant. Solutions have been proposed, but they have been slow to become accepted by device manufactures because they have required devices to be manufactured with secure clocks. Secure clocks have been VERY slow to become accepted because of their impact to the BOM.

This invention simplifies life greatly for users. This is the core advantage. After the production and engineering problems are eventually solved which they are close to being solved, an even larger problem looms on the horizon. The focus has been on putting a drm on the device instead of creating a usable device. Users are not going to like or understand the idea that they have clock restrictions instead of usage restrictions on their media consumption. This invention though, uses a model that users understand, are familiar with and have already accepted.

3. YOU MUST include at least one figure illustrating the invention. If the invention relates to software, include a flowchart or pseudo-code representation of the algorithm.

The following flow is imagined:

- Obtain a new device

- Register device with a DMS
- DMS sends a recharge token to the device
    - Numerous protocols are possible some of which are claims of this invention
    - Any communication medium should be usable: WiFi, WAN, GPRS, Lan, USB, 1394, SMS, MMS etc.
- Freely transfer subscription media to devices.
    - No protocol is required
    - Again any communication medium, examples are: WiFi, WAN, GPRS, Lan, USB, 1394, etc.
- Play the media consuming the intrinsic rights the device has
- At some point either automatically or at the users request, the device will be issued a refresh token.
- If the user refuses to refresh/recharge their device, does not connect their device to a DMS, cancels their subscription to their DMS, hard re-sets their device, etc, The device will refuse to playback the user's subscription content. Users will understand this because the device will behave as if the battery has run out on their device.

4. Value of your invention to RealNetworks (how will or could it be used?).

We will use this invention to partner with device manufactures and create the subscription service of the future. We can lock others out of using similar usable technology and give our subscription services a huge competitive edge.

5. Explain how your invention is novel. If the technology itself is not new, explain what makes it different.

No rights are transferred from the DMA to the device. Instead a simple refresh message is transferred and Devices are created in a less general but more single purposed format.

6. Identify the closest or most pertinent prior art that you are aware of from RealNetworks or anyone else.

Microsoft's PD protocol. See novelty of our approach for differentiation. Prior work we have done with Portable devices has centered on the idea of transferring rights.

Sony OMG, again they rely on a clock on their devices to expire content.

One-way protocols used to bind content to a device. No expiration of content is done on devices thus these protocols are not useful for subscription services that require content to expire at some point in the future.

Who is likely to want to use this invention or infringe the patent if one is obtained and how would infringement be detected?

Microsoft, Pressplay, music match, apple. Infringement should be easily detectable.

EXHIBIT B

Main Identity

From: "Alain Hamel" <ahamel@real.com>
To: <adamc@real.com>
Sent: Friday, June 20, 2003 2:15 PM
Attach: Orange.doc
Subject: FW: proposed pd protocol

-----Original Message-----
From: Rahul Agarwal [mailto:rahul@real.com]
Sent: Monday, June 16, 2003 4:31 PM
To: Rahul Agarwal; Josh Hug; Jeff Ayars
Cc: Alain Hamel; Xiaodong "(Sheldon)" Fu
Subject: Re: proposed pd protocol

here is the red-lined doc.

At 09:17 AM 6/16/2003 -0700, Rahul Agarwal wrote:
>looks good. a few typos. I will send out a red-lined doc shortly.
>
>How do the robustness requirements for Orange differ from CPRM?
>
>- Rahul
>PS: any story behind the 'Orange' name for this protocol?
>
>At 07:11 PM 6/13/2003 -0700, Josh Hug wrote:
>>All,
>>
>>I have attached the first draft of our proposed one way device
>>protocol. Alain & Sheldon put this together today. The protocol is
>>simple with the goals of: (a) speeding our time to market (b) creating
>>the best user experience possible (c) maintaining an acceptable level of
>>security.
>>
>>I belive this proposal does a good job balancing usability and
>>security. After a device is licensed via this protocol, protected music
>>can be transferred directly to the device without any modifications
whatsoever.
>>
>>This protocol was designed to work either on removable media (Palm) or a
>>device with internal storage (Archos).
>>
>>Josh
>

Orange Protocol

Version 0.1

By Alain Hamel, Sheldon Fu, and Josh Hug

1 Abstract

This document specifies the initial version of the Orange protocol. The protocol enables servers (PC's) to transfer protected media to trusted clients (music players) via a storage device possessing a hierarchal file system.

2 Introduction

The client device writes its certificate to a specified portion of the storage device (see section 3.0). The server once connected to the storage device (which may or may not be removable from the client device) will read the certificate from the storage (see section 3 for the specified location). The certificate can then be used to encrypt the user's key. The user's encrypted key will then be written to a database on the storage device (see section 4 for the format of the "drop off"

Deleted: 4

database). The client can optionally decrypt the user key and store it in non-volatile non-user accessible memory. The client shall not delete the key from the database, and a client may choose to not cache the user key, decrypting it for every power up.

3 Device Certificate

Each player device comes with a unique X.509 format certificate embedded. The certificate is in PEM format (base64 encoded DER) and is signed by RealNetworks. The signature algorithm is sha1WithRSAEncryption. The certificate contains device's 1024 bit RSA public key for encryption.

A player device should treat the certificate as opaque data. The private key shall be guarded with appropriate key hiding and tamper resistance techniques (see robustness section 9.0). When a device is attached to a new piece of storage (in the case of removable media), the device should copy its certificate to a specific directory on the storage media.

The directory and file name for the certificate would be: \orange\device.cer.

4 "Drop Off" Database

The drop off database is a flat binary file containing a series of sequential records. The records can be of the following three types:

```
typedef struct _CActivateUser
{
      ULONG32          size;
      e_struct_type    type;
      GUID             userID;
      unsigned char    encryptedKey[128];
} CActivateUser;

typedef struct _CDeactiveDeviceRequest
{
      ULONG32          size;
      e_struct_type    type;
      GUID             userID [16];
      unsigned char    requestId[16];
} CdeactiveDeviceRequest;

typedef struct _CDeactiveDeviceResponse
{
      ULONG32          size;
      e_struct_type    type;
      GUID             userID [16];
      unsigned char    requestId[16];
      unsigned char    responseSignature[128];
} CDeactiveDeviceResponse;
```

Deleted: responseSigniture

where GUID and e_struct_type are defined as:

```
typedef struct  _GUID
{
      ULONG32     Data1;
      UINT16      Data2;
      UINT16      Data3;
      UCHAR       Data4[ 8 ];
}GUID;

typedef enum
{
      e_media_UserKey;
      e_media_DeactiveDeviceRequest,
      e_media_DeactiveDeviceResponse
} e_struct_type;
```

The dropoff database is always located in the "\orange\dropoff.db" file.

The "encryptedKey" contained within the CActivateUser is the structure CencryptedKey encrypted with RSA using a 1024 bit key within the device certificate specified in section 3.0. CUserKey is defined as follows:

```
typedef _CUserKey
{
      unsigned char          key[16];
      unsigned ULONG32       magic_number;
      unsigned char          randomdata[108];
} CUserKey;
```

The magic_number is defined as 0x20204f4b.

The "responseSignature" contained within the CDeactiveDeviceResponse is a RSA (using the key defined in section 4) signed CResponseSignature. CResponseSignature is defined as follows:

Deleted: responseSigniture

Deleted: ResponseSigniture

Deleted: ResponseSigniture

```
typedef _ CResponseSignature
{
      unsigned char    hash[20];  // a SHA-1 hash of
                                  // CDeactiveDeviceResponse
      unsigned char    random_data[108];
} CResponseSignature;
```

Deleted: ResponseSigniture

Deleted: ResponseSigniture

The file has a header of the following format:

```
typedef _CFileHeader
{
      unsigned char          magic_number;
      ULONG32                header_size;
```

```
    UINT16              build_number;
    unsigned char       minor_version;
    unsigned char       major_version;
} CFileHeader;
```

The magic number is defined as 0x524e4442.

Note that all data is stored in network byte order.

5 Client Authorization Process

When the server connects to the storage device it checks for the client certificate in the location specified in section 3.0. It then validates the certificate and checks the certificate against its revocation list. If the client is revoked the server issues a CDeactiveDeviceRequest as specified in section 4.0 (note that a hacked device may not respond to such a request).

However if the certificate passes verification the server then checks the drop off database (again as specified in section 4.0) and checks to see if the current user is authorized on this client. If the current user is not authorized on the client the server checks to see if the current user has rights to activate a new client. If the user has sufficient rights a CActivateUser record is added to the database.

Upon connecting the storage device to the client hardware in the case of removable media, or activation of a device with non-removable media, the client hardware will scan the drop off database for new CActivateUser records. If any new records are found it will decrypt the records and store the userid's and keys within it's private storage area (non-user accessible ram or optionally in flash memory, see key handling requirements in section 8).

Deleted: ROM

On attempt to playback content that is not authorized on the client, the client should check the drop-off point to see if a new user key has been given.

6 Device Deactivation Process

When the user wishes to deactivate a device the server software adds a CDeactiveDeviceRequest to the drop off database. The user's count of authorized devices is NOT decremented on the server software at this point. The client should scan for new CActivateUser as was noted in section 5.0. The client will look for any CDeactiveDeviceRequest structures. If any are found it will remove the key/user pair from it's internal non-user accessible ram/flash, it will remove the CDeactiveDeviceRequest structure from the database. The Client next adds a CDeactiveDeviceResponse to the database.

Deleted: rom

The server should poll the database for any new CDeactiveDeviceResponse structures. It will check the signature, if the signature is valid, the server will decrement the user's count of authorized devices.

7 Compliance rules for Licensed Products

7.1 Applicability.

These rules are applicable to Licensed Products that have implemented playback functionality of Helix DRM Audio Content.

7.2 OBLIGATIONS REGARDING PERSISTENT STORAGE OF CONTENT

Licensed products shall be constructed such that all data is treated as all Helix DRM content may not, once decrypted, be stored except as a for the sole purpose of enabling the immediate consumption of content but which (a) does not persist materially after the content has been consumed and (b) is not stored in a way which permits copying or storing of such data for other purposes.

7.3 PERMITTED OUTPUTS

7.3.1 Generally.

As set forth in more detail below, a Licensed Product shall not pass Decrypted Helix DRM content, whether in digital or analog form, to an output except as permitted below.

7.3.2 Analog.

There are no prohibitions relating to analog audio outputs.

7.3.3 Digital.

Licensed Products shall not output Helix DRM Audio content in digital form.

Deleted: 8.0 Key Handling requirements

8 Robustness requirements

8.1 Overview

Participating clients shall be designed and manufactured so as to resist attempts to circumvent the functions of the specification as more specifically described below.

8.1.1 Defeating Functions and Features

Participating clients shall not include: (1) switches, buttons, jumpers or software equivalents, (2) traces that could be cut, or (3) features/functions (eg: service menus) which could be used to defeat any function of the specification.

The list of functions which can not be thusly compromised includes but is not limited to:

1. Protection of the decrypted content
2. Protection of the private key
3. Enforcement of any associated playback restriction information.

8.1.2 Keep Secrets

Participating Clients shall be designed and manufactured such that they shall resist attempts to discover or reveal device keys or secret intermediate calculated cryptographic values.

8.2 Robustness requirements of software implementation

Any portion of a participating client that utilizes, in software, the client private key and/or the decrypted user key should use any and all reasonable methods to frustrate efforts to obtain this key.

All software implementations must at a minimum include: (1) techniques of obfuscation to disguise, hamper and frustrate attempts to discover the approaches used to hide the use and value of the keys (2) self-checking of the integrity of its component parts and be designed to result in a failure to provide decryption in the event of unauthorized modification. As a minimum this means the usage of signed code.

Further, they may include, but are not limited to: (1) code encryption (2) execution in ring zero or supervisor mode (3) embodiment in a secure physical implementation that cannot be reasonably read.

8.3 Robustness requirements of hardware implementation

Any hardware component that utilizes the client private key or users key, or that relies upon hardware to satisfy these robustness rules shall: (1) embed Highly Confidential Information in silicon circuitry or firmware which cannot reasonably be read (2) be designed such that attempts to remove or replace hardware elements in a way that would compromise the Highly Confidential Information would pose a serious risk of damaging the Participating Device. By way of example, a component which is soldered rather than socketed may be appropriate for these means.

8.3.1 Hardware Data paths

Decrypted data shall not be present on any user accessible buses in useable form. For this purpose user accessible bus is defined as: (1) an internal analogue connector which is designed for end user upgrades or readily facilitates end user access (2) a data bus that is designed for end user upgrades or access, eg: PCMCIA, CardBus, PCI bus.

A user accessible bus does not include memory buses, CPU buses or similar portions of a devices internal architecture.

8.4 Required Levels of Robustness

The levels of robustness described in sections throughout this section shall be implemented so that it is reasonably certain that they cannot be defeated or circumvented using "widely available tools" or "specialized Tools". And that only with great difficulty they can be circumvented with "professional tools."

8.4.1 Widely Available Tools

Widely Available Tools is defined as general-purpose tools or equipment that are widely available at a reasonable price, such as screwdrivers, jumpers, clips, file editors, and soldering irons.

8.4.2 Specialized Tools

Specialized Tools is defined as specialized electronic tools that are widely available at a reasonable price, such as memory readers and writers, debuggers, decompilers, or similar software development products.

8.4.3 Professional Tools

Professional Tools is defined as professional tools or equipment, such as logic analyzers, chip disassembly systems, or in circuit emulators.

EXHIBIT C

Subject: spec for user license request
Date: Tue, 01 Jul 2003 14:43:28 -0700
From: Qiang Luo <qluo@real.com>
To: Adam Cappio <adamc@real.com>, Sheldon Fu <sfu@real.com>, Josh Hug <jhug@real.com>, Alain Hamel <ahamel@real.com>, Brent Newman <bnewman@real.com>, Rahul Agarwal <rahul@real.com>

When making machineLicense request to the license server, the storefront shall format it as a subscription request where Subscription_SubscriptionGUID=UserGUID.

There will be one additional parameter, Subscription_ContentKey=userKey. The userKey is retrieved from the licensed user's account. The user key is encrypted with the packager's public key.

I propose that we make Subscription_Title=machineLicense mandatory for all machineLicense.

Sheldon asked me to design a new interface to get a list of all userGUIDs that are licensed on the PC. This is required in the Orange protocol. We should be able to easily implement this with GetAllRecordInfo(RT_SUBSCRIPTION) call then matching each subscription's title for "machineLicense".

Qiang

EXHIBIT D

Main Identity

From: "Josh Hug" <jhug@real.com>
To: "Darryl Wood" <darrylwood@real.com>; "Jeff Watts" <jeffwatts@real.com>; "Timothy Graham" <tgraham@real.com>
Cc: "adam Cappio" <adamc@real.com>; "Alain Hamel" <ahamel@real.com>
Sent: Wednesday, July 16, 2003 2:14 PM
Attach: Orange v.12.doc
Subject: orange protocol.

Hey guys,

I hope I didn't get us too side tracked on the Activation/de-activation discussions today. I have attached our protocol for key transfer to devices. It explains the messages we will need to send from the DRM to the device.

Jos

Orange Protocol

Version 0.11

By Alain Hamel, Sheldon Fu, and Josh Hug

1 Abstract

This document specifies the initial version of the Orange protocol. The protocol enables servers (PC's) to transfer protected media to trusted clients (music players) via a storage device possessing a hierarchal file system.

2 Introduction

The client device writes its certificate to a specified portion of the storage device (see section 3.0). The server once connected to the storage device (which may or may not be removable from the client device) will read the certificate from the storage (see section 3 for the specified location). The certificate can then be used to encrypt the user's key. The user's encrypted key will then be written to a database on the storage device (see section 3 for the format of the "drop off"

database). The client can optionally decrypt the user key and store it in non-volatile non-user accessible memory. The client shall not delete the key from the database, and a client may choose to not cache the user key, decrypting it for every power up.

3 Device Certificate

Each player device comes with a unique X.509 format certificate embedded. The certificate is in PEM format (base64 encoded DER) and is signed by RealNetworks. The signature algorithm is sha1WithRSAEncryption. The certificate contains device's 1024 bit RSA public key for encryption.

A player device should treat the certificate as opaque data. The private key shall be guarded with appropriate key hiding and tamper resistance techniques (see robustness section 9.0). When a device is attached to a new piece of storage (in the case of removable media), the device should copy its certificate to a specific directory on the storage media.

The directory and file name for the certificate would be: \orange\device.cer.

4 "Drop Off" Database

The drop off database is a flat binary file containing a series of sequential records. The records can be of the following three types:

```
typedef struct _CActivateUser
{
      ULONG32          size;
      e_struct_type    type;
      GUID             userID;
      unsigned char    encryptedKey[128];
} CActivateUser;

typedef struct _CDeactiveDeviceRequest
{
      ULONG32          size;
      e_struct_type    type;
      GUID             userID [16];
      unsigned char    requestId[16];
} CdeactiveDeviceRequest;

typedef struct _CDeactiveDeviceResponse
{
      ULONG32          size;
      e_struct_type    type;
      GUID             userID [16];
      unsigned char    requestId[16];
      unsigned char    responseSignature[128];
} CDeactiveDeviceResponse;
```

where GUID and e_struct_type are defined as:

```
typedef struct  _GUID
{
      ULONG32     Data1;
      UINT16      Data2;
      UINT16      Data3;
      UCHAR       Data4[ 8 ];
}GUID;

typedef enum
{
      e_media_UserKey;
      e_media_DeactiveDeviceRequest,
      e_media_DeactiveDeviceResponse
} e_struct_type;
```

The dropoff database is always located in the "\orange\dropoff.db" file.

The "encryptedKey" contained within the CActivateUser is the structure CencryptedKey encrypted with RSA using a 1024 bit key within the device certificate specified in section 3.0. CUserKey is defined as follows:

```
typedef _CUserKey
{
      unsigned char          key[16];
      unsigned ULONG32       magic_number;
      unsigned char          randomdata[108];
} CUserKey;
```

The magic_number is defined as 0x20204f4b.

The "responseSignature" contained within the CDeactiveDeviceResponse is a RSA (using the key defined in section 4) signed CResponseSignature. CResponseSignature is defined as follows:

```
typedef _ CResponseSignature
{
      unsigned char    hash[20];  // a SHA-1 hash of
                                  // CDeactiveDeviceResponse
      unsigned char    random_data[108];
} CResponseSignature;
```

The file has a header of the following format:

```
typedef _CFileHeader
{
      unsigned char          magic_number;
      ULONG32                header_size;
```

```
    UINT16                  build_number;
    unsigned char           minor_version;
    unsigned char           major_version;
} CFileHeader;
```

The magic number is defined as 0x524e4442.

Note that all data is stored in network byte order.

5 Client Authorization Process

When the server connects to the storage device it checks for the client certificate in the location specified in section 3.0. It then validates the certificate and checks the certificate against its revocation list. If the client is revoked the server issues a CDeactiveDeviceRequest as specified in section 4.0 (note that a hacked device may not respond to such a request).

However if the certificate passes verification the server then checks the drop off database (again as specified in section 4.0) and checks to see if the current user is authorized on this client. If the current user is not authorized on the client the server checks to see if the current user has rights to activate a new client. If the user has sufficient rights a CActivateUser record is added to the database.

Upon connecting the storage device to the client hardware in the case of removable media, or activation of a device with non-removable media, the client hardware will scan the drop off database for new CActivateUser records. If any new records are found it will decrypt the records and store the userid's and keys within it's private storage area (non-user accessible ram or optionally in flash memory, see robustness rules in section 8).

On attempt to playback content that is not authorized on the client, the client should check the drop-off point to see if a new user key has been given.

6 Client Deactivation Process

When the user wishes to deactivate a device the server software adds a CDeactiveDeviceRequest to the drop off database. The user's count of authorized devices is NOT decremented on the server software at this point. The client should scan for new CActivateUser as was noted in section 5.0. The client will look for any CDeactiveDeviceRequest structures. If any are found it will remove the key/user pair from it's internal non-user accessible ram/flash, it will remove the CDeactiveDeviceRequest structure from the database. The Client next adds a CDeactiveDeviceResponse to the database.

The server should poll the database for any new CDeactiveDeviceResponse structures. It will check the signature, if the signature is valid, the server will decrement the user's count of authorized devices.

7 Compliance rules for Licensed Products

7.1 Applicability.

These rules are applicable to Licensed Products that have implemented playback functionality of Helix DRM Audio Content.

7.2 OBLIGATIONS REGARDING PERSISTENT STORAGE OF CONTENT

Licensed products shall be constructed such that all data is treated as all Helix DRM content may not, once decrypted, be stored except as a for the sole purpose of enabling the immediate consumption of content but which (a) does not persist materially after the content has been consumed and (b) is not stored in a way which permits copying or storing of such data for other purposes.

7.3 PERMITTED OUTPUTS

7.3.1 Generally.

As set forth in more detail below, a Licensed Product shall not pass Decrypted Helix DRM content, whether in digital or analog form, to an output except as permitted below.

7.3.2 Analog.

There are no prohibitions relating to analog audio outputs.

7.3.3 Digital.

Licensed Products shall not output Helix DRM Audio content in digital form.

8 Robustness requirements

8.1 Overview

Participating clients shall be designed and manufactured so as to resist attempts to circumvent the functions of the specification as more specifically described below.

8.1.1 Defeating Functions and Features

Participating clients shall not include: (1) switches, buttons, jumpers or software equivalents, (2) traces that could be cut, or (3) features/functions (eg: service menus) which could be used to defeat any function of the specification.

The list of functions which can not be thusly compromised includes but is not limited to:

1. Protection of the decrypted content
2. Protection of the private key
3. Enforcement of any associated playback restriction information.

8.1.2 Keep Secrets

Participating Clients shall be designed and manufactured such that they shall resist attempts to discover or reveal device keys or secret intermediate calculated cryptographic values.

8.2 Robustness requirements of software implementation

Any portion of a participating client that utilizes, in software, the client private key and/or the decrypted user key should use any and all reasonable methods to frustrate efforts to obtain this key.

All software implementations must at a minimum include: (1) techniques of obfuscation to disguise, hamper and frustrate attempts to discover the approaches used to hide the use and value of the keys (2) self-checking of the integrity of its component parts and be designed to result in a failure to provide decryption in the event of unauthorized modification. As a minimum this means the usage of signed code.

Further, they may include, but are not limited to: (1) code encryption (2) execution in ring zero or supervisor mode (3) embodiment in a secure physical implementation that cannot be reasonably read.

8.3 Robustness requirements of hardware implementation

Any hardware component that utilizes the client private key or users key, or that relies upon hardware to satisfy these robustness rules shall: (1) embed Highly Confidential Information in silicon circuitry or firmware which cannot reasonably be read (2) be designed such that attempts to remove or replace hardware elements in a way that would compromise the Highly Confidential Information would pose a serious risk of damaging the Participating Device. By way of example, a component which is soldered rather than socketed may be appropriate for these means.

8.3.1 Hardware Data paths

Decrypted data shall not be present on any user accessible buses in useable form. For this purpose user accessible bus is defined as: (1) an internal analogue connector which is designed for end user upgrades or readily facilitates end user access (2) a data bus that is designed for end user upgrades or access, eg: PCMCIA, CardBus, PCI bus.

A user accessible bus does not include memory buses, CPU buses or similar portions of a devices internal architecture.

8.4 Required Levels of Robustness

The levels of robustness described in sections throughout this section shall be implemented so that it is reasonably certain that they cannot be defeated or circumvented using "widely available tools" or "specialized Tools". And that only with great difficulty they can be circumvented with "professional tools."

8.4.1 Widely Available Tools

Widely Available Tools is defined as general-purpose tools or equipment that are widely available at a reasonable price, such as screwdrivers, jumpers, clips, file editors, and soldering irons.

8.4.2 Specialized Tools

Specialized Tools is defined as specialized electronic tools that are widely available at a reasonable price, such as memory readers and writers, debuggers, decompilers, or similar software development products.

8.4.3 Professional Tools

Professional Tools is defined as professional tools or equipment, such as logic analyzers, chip disassembly systems, or in circuit emulators.

EXHIBIT E

Subject:replay attack on Orange protocol and solutions
Date: Fri, 1 Aug 2003 15:47:52 -0400
From: Sheldon Fu <sfu@real.com>
To:'Josh Hug' <jhug@real.com>, Alain Hamel <ahamel@real.com>, Qiang Luo <qluo@real.com>
CC: Adam Cappio <adamc@real.com>, Rahul Agarwal <rahul@real.com>

Josh pointed out that our current Orange protocol is vulnerable to replay attack --- somebody could record the 'ActivateUser' dropoff DB and later after the device is de-registered, resend the saved DB to the device to activate the device again without PC DRM knowing anything about it.

The prevention of replay attack seems not an easy task for the device:

1. Since device may not have clock/timer and we could use mass storage as communication channel, time-out check is out.
2. Same as in the PC license insertion replay prevention; we could have PC generate a unique record ID (or DB ID) each time and let device remember is a list of say last '100' IDs.
3. Alternatively, we could use a counter in DB header so that each time anybody writes to it, increases the counter and both PC and device rememebers the 'last seen counter'. This means that the dropoff DB has to be always there and can not be deleted at any time.

Big problem: Both (2) and (3) have problem when a device is reset so we will forget the list or the 'last seen counter'. It is so easy to reset a device, this is really big problem.

We could improve by (3):

4. Require that only the device could create the DB, and each time the device create a DB (after reset, or not after reset), it has to put a RANDOM number as the initial counter value. Big drawback is that it may be really tough for a device without any clock and persistence storage other than flash (like Palm) to generate a random number after reset.

Anybody has a better idea? Or we should just spec (3) and (4) in Orange and let device developers to scratch their heads to come out a random number? For a lame (and 'fun' ;-)) solution, they could ask the user to punch a button 20 times then measure the intervals between key events to come out a random number, each time a dropoff DB needs to be created.

fxd

EXHIBIT F

Main Identity

From: "Sheldon Fu" <sfu@real.com>
To: "'Qiang Luo'" <qluo@real.com>; "'Josh Hug'" <jhug@real.com>; "'Alain Hamel'" <ahamel@real.com>
Cc: "'Adam Cappio'" <adamc@real.com>; "'Rahul Agarwal'" <rahul@real.com>
Sent: Saturday, August 02, 2003 7:50 PM
Attach: Orange v.14a.doc
Subject: RE: replay attack on Orange protocol and solutions

Good point. Comment inline. Review the new spec please.

fxd

> -----Original Message-----
> From: Qiang Luo [mailto:qluo@real.com]
> Sent: Saturday, August 02, 2003 4:35 PM
> To: Sheldon Fu; 'Josh Hug'; 'Alain Hamel'
> Cc: 'Adam Cappio'; 'Rahul Agarwal'
> Subject: RE: replay attack on Orange protocol and solutions
>
> The current header structure and the new protection scheme will not work
> if
> the encryption protocol is RSA. The PC server can encrypt and write to
> DB. But it can not decrypt, update, and then write encrypted_challenge to
> DB without PD's private key.
>
> As a result, PC side can not decrypt the encrypted_challenge to get and
> verify the hash. When PC writes the encrypted_challenge, it will destroy
> the DB's access number.

Good catch. I overlooked that. This will be a major difficulty. We have to change the protocol to a somehow ugly solution I think. Let's do this:

PD always put a signed challenge in the header. PC will verify it (and get the challenge bytes). We will change the PC side of CUserKey field in the CActivateUser record to include the challenge bytes from header. Device will certainly need to verify the challenge byte also when it reads the CUserKey fields. This is not too bad. we actually save a hash operation.

This will work just because we don't care about replay attack of 'deactivation user'. That's suicide for the hacker. ;-)

For symmetric version, we will change to use the same scheme to be

consistent.

See the attached new document to review the updated version. We could use a full signature for the challenge data in the header, but I don't think that's really necessary, unless somebody can point me to some security risk of proposed method.

>
> The DB access number, challenge_data, belongs to the PD client. PC
> doesn't
> and needs not to know about it. PC, however, needs to verify the hash
to
> prevent a hacker from sending to PC the PD's CDeactiveDeviceResponse
with
> a
> de-activated user record added back after pull the card from the PD.

No. This shouldn't be a problem. DeactiveUser has its own challenge/response. It is not subject to replay attack (from PC's point of view), unless I (or Al) missed something. The new replay attack protection is really just for ActivateUser record. Detail of your concern?

>
> The following change to the structure will take care of this I think:
> typedef _CFileHeader
> {
> unsigned char magic_number;
> ULONG32 header_size;
> UINT16 build_number;
> unsigned char minor_version;
> unsigned char major_version;
> unsigned char db_hash[20];
> unsigned char encrypted_challenge[128];
> } CFileHeader;
> where the db_hash holds the SHA-1 hash of the encrypted_challenge and
the
> DB records.
>
> Now the problem is how do we prevent the hacker from tampering the DB
and
> then patching up the clear db_hash? We seem to need some kind of
shared
> secret to sign/encrypt it. Any ideas?

No, we don't. In the case of RSA, device knows absolutely no secret of PC DRM. That won't work.

>
> Qiang
>

> At 09:14 PM 8/1/2003 -0400, Sheldon Fu wrote:
>
> >Initial cut on the challenge method for replay attack prevention. I added
> >header data structure and explanation. Josh said he will add the detailed
> >text about requirement and suggestion on how to implement the challenge
> >which will be highly device dependent:
> >
> >
> > * Device with non-volatile secret storage (that can be written to
> > frequently): challenge could be simply a number start at zero and
> > increased each time the dropoff DB is used.
> > * Device CPU support random generation: challenge could be a random
> > number each time dropoff DB is used.
> > * No easy randgen, no non-volatile secret storage: Try hard to
> > generate a random number at reset, and increase it each time the dropoff
> > DB is used.
> >
> >
> >Have a nice weekend.
> >
> >
> >
> >fxd
> >
> >
> >
> >-----Original Message-----
> >From: Josh Hug [mailto:jhug@real.com]
> >Sent: Friday, August 01, 2003 7:50 PM
> >To: Sheldon Fu; 'Alain Hamel'; 'Qiang Luo'
> >Cc: 'Adam Cappio'; 'Rahul Agarwal'
> >Subject: RE: replay attack on Orange protocol and solutions
> >
> >
> >
> >At 04:01 PM 8/1/2003, Sheldon Fu wrote:
> >
> >-----Original Message-----
> >From: Josh Hug [mailto:jhug@real.com]
> >Sent: Friday, August 01, 2003 6:43 PM
> >To: Sheldon Fu; 'Alain Hamel'; 'Qiang Luo'
> >Cc: 'Adam Cappio'; 'Rahul Agarwal'
> >Subject: RE: replay attack on Orange protocol and solutions
> >but it is more tricky for a device that has NO private non-volatile
> >storage except the db. then we really are relying on the device to get

> us
> >a "random" number.
> >
> >
> >
> >Right, for built-in storage device iPod-like, yes they should have
area
> in
> >non-volatile mass storage we could use (and hopefully not user
> >accessible). But for players with external cards as storage, most
likely
> >it won t have (or not want us to have) any non-volatile storage that
can
> >be regularly written to. I hope we are not thinking about writing to
the
> >same physical flash chip that holds the device firmware in any case.
> Those
> >places are not meant to be written too frequently.
> >
> >
> >agreed. we can't write to the device firmware.
> >
> >
> > And for orange deactivation to work, PC already has to remember the
> > random challenge in the record, it might as well to remember the DB
> > challenge number too.
> >
> >
> >but the db challenge would have to be remembered across the lift time
of
> a
> >Devices interaction w/ a PC? what happens when the device connects
to
> >another PC?
> >
> >
> >
> >Good point. Along the same line of thinking about remember things ,
what
> >if the memory card is plugged into a different device or multiple
card is
> >used? For MMC/SD, device will have to use the ID to index the
challenge
> >and still not all memory cards have IDs. If we assume there is an ID,
PC
> >can do the same.
> >
> >
> >but also for the palm device w/ no internal memory will loose all
their
> >rights after they are re-set. how about we generate the initial

random
> >value on the PC w/ the initial image we send over? (then the device
> >increments it from then on).
> >
> >in the case that the device is "restored" after a re-set, then we are
> fine
> >because the computer will have backed up the memory w/ the device id.
> >
> >
> >
> > Are we really targeting the devices with built-in storage now?
> >
> >Maybe we should define different classes of device first then tailor
the
> >protocol for each of them.
> >
> >
> >Yes, we will have to have a tailored protocol for each class of
> >device. We need to nail down our protocol for two devices ASAP.
> >a.) the Creative device
> >b.) the Nokia device
> >
> >These should be similar. Both have internal storage, so we should be
> able
> >to store our current random "seed" on the internal storage. They
have a
> >bit of a different way of passing data (nokia drop box, creative -
the
> >creative protocol.)
> >
> >We will need a variant of this protocol for Palm device, maybe w/ the
> >initial random number generated on the PC..
> >
> >and then a variant of the protocol for the Archos device as well
(maybe
> >hiding the seed on the HD...)
> >
> >Josh

Orange Protocol

Version 0.13

By Alain Hamel, Sheldon Fu, and Josh Hug

Deleted: 4
Deleted: 4
Deleted: 5
Deleted: 5
Deleted: 5
Deleted: 5
Deleted: 5
Deleted: 5
Deleted: 5
Deleted: 5
Deleted: 5
Deleted: 6
Deleted: 6
Deleted: 6
Deleted: 6
Deleted: 7
Deleted: 7
Deleted: 7
Deleted: 7

1 Abstract

This document specifies the initial version of the Orange protocol. The protocol enables servers (PC's) to transfer protected media to trusted clients (music players) via a storage device possessing a hierarchal file system.

Formatted: Bullets and Numbering

2 There are two variants of the protocol each with different security characteristics. They vary in their application of cryptographic protocols, asymmetric vs. symmetric.Introduction

Deleted: ¶

The client device writes its certificate to a specified portion of the storage device (see section 3.0). The server once connected to the storage device (which may

or may not be removable from the client device) will read the certificate from the storage (see section 3 for the specified location). The certificate can then be used to encrypt the user's key. The user's encrypted key will then be written to a database on the storage device (see section 3 for the format of the "drop off" database). The client can optionally decrypt the user key and store it in non-volatile non-user accessible memory. The client shall not delete the key from the database, and a client may choose to not cache the user key, decrypting it for every power up.

The best version uses asymmetric cryptography to transfer keys from a server to a trusted client. Because each unit is individualized, this version provides individualized revocation of devices. Each device uses their own key pair, so hacking one device does not compromise the rest of the clients in the field.

The second version uses symmetric cryptography. The symmetric version is targeted at devices that do not have the processing power to run RSA. There are certain classes of secure content with lower security requirements (e.g. owned content). Demand for content with a lower security requirement on clients without resources to do asymmetric cryptography lead to the creation of a symmetric cryptographic method.

Deleted: ¶

3 Device Certificate

Each player device comes with a unique X.509 format certificate embedded. The certificate is in PEM format (base64 encoded DER) and is signed by RealNetworks. The signature algorithm is sha1WithRSAEncryption.

The certificate will be slightly different depending on whether the implementation is the asymmetric or symmetric protocol variant. The asymmetric certificate will contain device's 1024 bit RSA public key for encryption. The symmetric certificate will include a device class identifier, so the server can associate the certificate with the appropriate secret.

Deleted: s

A player device should treat the certificate as opaque data. The private or secret key shall be guarded with appropriate key hiding and tamper resistance techniques (see robustness section 9.0). When a device is attached to a new piece of storage (in the case of removable media), the device should copy its certificate to a specific directory on the storage media.

The directory and file name for the certificate would be: \helix_d\device.cer.

Deleted: orange

4 "Drop Off" Database

The drop off database is a flat binary file containing a series of sequential records. The records can be of the following three types:

```
typedef struct _CActivateUser
```

```
{
     ULONG32          size;
     e_struct_type    type;
     GUID             userID;
     unsigned char    encryptedKey[128];
} CActivateUser;

typedef struct _CDeactiveDeviceRequest
{
     ULONG32          size;
     e_struct_type    type;
     GUID             userID [16];
     unsigned char    requestId[16];
} CdeactiveDeviceRequest;

typedef struct _CDeactiveDeviceResponse
{
     ULONG32          size;
     e_struct_type    type;
     GUID             userID [16];
     unsigned char    requestId[16];
     unsigned char    responseSignature[128];
} CDeactiveDeviceResponse;
```

where GUID and e_struct_type are defined as:

```
typedef struct  _GUID
{
     ULONG32    Data1;
     UINT16     Data2;
     UINT16     Data3;
     UCHAR      Data4[ 8 ];
}GUID;

typedef enum
{
     e_media_UserKey;
     e_media_DeactiveDeviceRequest,
     e_media_DeactiveDeviceResponse
} e_struct_type;
```

The dropoff database is always located in the "\helix_d\dropoff.db" file.

Deleted: orange

The "encryptedKey" contained within the CActivateUser is the structure UserKey encrypted with RSA using a 1024 bit key within the device certificate specified in section 3.0. In the case of the symmetric protocol, CUserKey is encrypted with the server's shared symmetric secrete. CUserKey is defined as follows:

Deleted: Cencrypted

```
typedef _CUserKey
{
```

```
    unsigned char          key[16];
    unsigned ULONG32       magic_number;
    unsigned char          challenge[16];
    unsigned char          randomdata[108];
} CUserKey;
```

The magic_number is defined as 0x20204f4b. The challenge field is taken from the DB header "challenge". Device will verify both magic_number and challenge to validate the CUserKey record.

The "responseSignature" contained within the CDeactiveDeviceResponse is a RSA (using the key defined in section 4) signed CResponseSignature. The secret key is used instead in the case of the symmetric version of the protocol. CResponseSignature is defined as follows:

```
typedef _ CResponseSignature
{
    unsigned char    hash[20];  // a SHA-1 hash of
                                // CDeactiveDeviceResponse
    unsigned char    random_data[108];
} CResponseSignature;
```

The file has a header of the following format:

```
typedef _CFileHeader
{
    ULONG32            magic_number;
    ULONG32            header_size;
    UINT16             build_number;
    unsigned char      minor_version;
    unsigned char      major_version;
    unsigned char      signed_challenge[128];
} CFileHeader;
```

Deleted: unsigned char

The magic number is defined as 0x524e4442.

Note that all data is stored in network byte order.

The signed_challenge is signed/encrypted using the RSA private key or shared symmetric secret hold by the device. Its internal structure is defined as:

```
Typedef   CChallenge
{
    unsigned char      challenge[16];
    ULONG32            magic_number;
    unsigned char      randomdata[127-16-16];
} CChallenge;
```

The magic_number is defined to be 0xDEADBEEF.

challenge_data is a 16 byte binary data maintained by the device. It should be changed to a new value by the device for each dropoff DB read/write operation. Device should internally maintain a copy of the current challenge_data and always compare two values to verify that dropoff DB is a current one, instead of a previously seen one to prevent replay attack. see [TBD] section for the robustness requirement on challenge_data and possible implementation.

Device must initialize at least the dropoff DB header, before the communication between the device and PC can start. PC will not create a dropoff DB if one is not present.

Deleted: ¶

5 Client Authorization Process

When the server connects to the storage device it checks for the client certificate in the location specified in section 3.0. It then validates the certificate and checks the certificate against its revocation list. If the client is revoked the server issues a CDeactiveDeviceRequest as specified in section 4.0 (note that a hacked device may not respond to such a request).

However if the certificate passes verification the server then checks the drop off database (again as specified in section 4.0) and checks to see if the current user is authorized on this client. If the current user is not authorized on the client the server checks to see if the current user has rights to activate a new client. If the user has sufficient rights a CActivateUser record is added to the database.

Upon connecting the storage device to the client hardware in the case of removable media, or activation of a device with non-removable media, the client hardware will scan the drop off database for new CActivateUser records. If any new records are found it will decrypt the records and store the userid's and keys within it's private storage area (non-user accessible ram or optionally in flash memory, see robustness rules in section 8).

On attempt to playback content that is not authorized on the client, the client should check the drop-off point to see if a new user key has been given.

6 Client Deactivation Process

When the user wishes to deactivate a device the server software adds a CDeactiveDeviceRequest to the drop off database. The user's count of authorized devices is NOT decremented on the server software at this point. The client should scan for new CActivateUser as was noted in section 5.0. The client will look for any CDeactiveDeviceRequest structures. If any are found it will remove the key/user pair from it's internal non-user accessible ram/flash, it will

remove the CDeactiveDeviceRequest structure from the database. The Client next adds a CDeactiveDeviceResponse to the database.

The server should poll the database for any new CDeactiveDeviceResponse structures. It will check the signature, if the signature is valid, the server will decrement the user's count of authorized devices.

7 Compliance rules for Licensed Products

7.1 Applicability.

These rules are applicable to Licensed Products that have implemented playback functionality of Helix DRM Audio Content.

7.2 OBLIGATIONS REGARDING PERSISTENT STORAGE OF CONTENT

Licensed products shall be constructed such that all data is treated as all Helix DRM content may not, once decrypted, be stored except as a for the sole purpose of enabling the immediate consumption of content but which (a) does not persist materially after the content has been consumed and (b) is not stored in a way which permits copying or storing of such data for other purposes.

7.3 PERMITTED OUTPUTS

7.3.1 Generally.

As set forth in more detail below, a Licensed Product shall not pass Decrypted Helix DRM content, whether in digital or analog form, to an output except as permitted below.

7.3.2 Analog.

There are no prohibitions relating to analog audio outputs.

7.3.3 Digital.

Licensed Products shall not output Helix DRM Audio content in digital form.

Deleted: ¶

8 Robustness requirements

8.1 Overview

Participating clients shall be designed and manufactured so as to resist attempts to circumvent the functions of the specification as more specifically described below.

8.1.1 Defeating Functions and Features

Participating clients shall not include: (1) switches, buttons, jumpers or software equivalents, (2) traces that could be cut, or (3) features/functions (eg: service menus) which could be used to defeat any function of the specification.

The list of functions which can not be thusly compromised includes but is not limited to:

1. Protection of the decrypted content
2. Protection of the private key
3. Enforcement of any associated playback restriction information.

8.1.2 Keep Secrets

Participating Clients shall be designed and manufactured such that they shall resist attempts to discover or reveal device keys or secret intermediate calculated cryptographic values.

8.2 Robustness requirements of software implementation

Any portion of a participating client that utilizes, in software, the client private key and/or the decrypted user key should use any and all reasonable methods to frustrate efforts to obtain this key.

All software implementations must at a minimum include: (1) techniques of obfuscation to disguise, hamper and frustrate attempts to discover the approaches used to hide the use and value of the keys (2) self-checking of the integrity of its component parts and be designed to result in a failure to provide decryption in the event of unauthorized modification. As a minimum this means the usage of signed code.

Further, they may include, but are not limited to: (1) code encryption (2) execution in ring zero or supervisor mode (3) embodiment in a secure physical implementation that cannot be reasonably read.

8.3 Robustness requirements of hardware implementation

Any hardware component that utilizes the client private key or users key, or that relies upon hardware to satisfy these robustness rules shall: (1) embed Highly Confidential Information in silicon circuitry or firmware which cannot reasonably be read (2) be designed such that attempts to remove or replace hardware elements in a way that would compromise the Highly Confidential Information would pose a serious risk of damaging the Participating Device. By way of example, a component which is soldered rather than socketed may be appropriate for these means.

8.3.1 Hardware Data paths

Decrypted data shall not be present on any user accessible buses in useable form. For this purpose user accessible bus is defined as: (1) an internal analogue connector which is designed for end user upgrades or readily facilitates end user

access (2) a data bus that is designed for end user upgrades or access, eg: PCMCIA, CardBus, PCI bus.

A user accessible bus does not include memory buses, CPU buses or similar portions of a devices internal architecture.

8.4 Required Levels of Robustness.

The levels of robustness described in sections throughout this section shall be implemented so that it is reasonably certain that they cannot be defeated or circumvented using "widely available tools" or "specialized Tools". And that only with great difficulty they can be circumvented with "professional tools."

8.4.1 Widely Available Tools

Widely Available Tools is defined as general-purpose tools or equipment that are widely available at a reasonable price, such as screwdrivers, jumpers, clips, file editors, and soldering irons.

8.4.2 Specialized Tools

Specialized Tools is defined as specialized electronic tools that are widely available at a reasonable price, such as memory readers and writers, debuggers, decompilers, or similar software development products.

8.4.3 Professional Tools

Professional Tools is defined as professional tools or equipment, such as logic analyzers, chip disassembly systems, or in circuit emulators.

EXHIBIT G

Main Identity

From: "Josh Hug" <jhug@real.com>
To: "Rahul Agarwal" <rahul@real.com>; "Xiaodong (Sheldon) Fu" <sfu@real.com>; "Brent Newman" <bnewman@real.com>; "Qiang Luo" <qluo@real.com>; "Adam Cappio" <adamc@real.com>
Sent: Thursday, August 07, 2003 3:30 AM
Attach: helix device DRM.vsd; Helix Device DRM v.21.doc
Subject: Helix Device DRM v.21.doc

Ok... here is the latest revision. getting closer, but still needs more work.

we REALLY need to have this pretty final by the end of this week. It still has quite a bit more work to be done on it. I have a very busy day tomorrow and am going to be out on Friday. Therefore it would be best if I could get some help?

Can I take, Volunteers to help out editing the sections and expanding on areas?

Adam, can you take over consolidating and handle the master?

I am not married to current structure, open to suggestions/changes if you have better organization ideas?

I am pretty comfortable with sections 1-4 as they are now. Section 5 is hit and miss...

Note that I changed a few of the structure names to be consistent. Ramifications on the code i know... Hopefully we aren't too late, this should be the final name change round!

TODO:

Section 4.2:
Sheldon to update to reflect the fact we will use our own certificate format and the symmetric certificates will contain the asymmetrically encrypted user key.

Section 5.1 (registration)
needs edited, expanding upon.
need to separate registration process on PC from registration process on the Device.
i have updated the structures section.

Section 5.2 (de-registration)
Needs edited, expanded upon.
needs structures updated explained.

needs security review: we are still ok right? need to reconcile changes vs. the way Al originally did it. he signed the response (w/ a hash of the db.) instead of encrypting the request... he might have been thinking better than I am with reversing it. (but we shouldn't sign w/ the same key, so it is better to encrypt if there is no security difference). well... i guess by signing the info, the PC doesn't have to remember what the original request id was?? no... it still needs to.

Section 6.

Needs appropriate text takend from 5.1 & 5.2 and new text created to define the on device behavior.

Appendix A.

Needs to be filled in. Order of importance is (1) Creative case, (2) Palm, (3) Nokia & Archos

Josh

Helix Device DRM Protocol

Version 0.21

1. Abstract

This document specifies the initial version of the Helix Device DRM protocol. The protocol enables servers (PCs) to transfer protected media to trusted clients (PDs). PDs are free to expose themselves to the OS in numerous ways. Two common methods are using a hierarchal file system accessible as a mass storage device or using custom device drivers.

The protocol varies in its use of cryptographic asymmetric vs. symmetric cryptography. The asymmetric protocol is designed for devices that have an individualized private key. The downside of the asymmetric protocol is this individualization can be burden on manufacturing. Therefore, we have created a symmetric protocol variant. Both methods provide for device revocation and use an individualized transfer key. The difference is where the individualization occurs. The symmetric key variant does individualization device initialization/key transfer time whereas the asymmetric version individualizes the keys at manufacturing time.

The time at which individualization occurs is the chief difference between the two protocols. In general it is preferred to have individualization done at manufacturing time, but the costs associated can be prohibitive. Therefore an alternative allowing for dynamic individualization is provided.

2. Background

The Helix Device DRM protocol is designed to authorize devices to consume User licensed content.

2.1 PC Experience

User licensed content is personalized to the Licensed User at the time of download. The Content contains information about the User to which the content is licensed to, and also contains the encrypted Content-Key required to decrypt the content during playback. The Content Provider assigns a unique User-Key to each Licensed User. The User-Key is used to encrypt the Content-Key in every personalized content file. Once the User has "registered" their machine with the Content Provider, they receive a License with the User-Key and associated Rights. The User License is kept in a Secure Vault on the machine and is tied to the machine and to the User. Access to the User License enables access to User's entire content catalog.

If the User wishes to use their content catalog on a different machine, all they need to do is copy their content to the new machine and get a License with the User-Key by performing a "registration" of their new machine with the Content Provider. The Content Provider may allow the User to play back their content on

up-to "N" registered machines. If the User has already registered "N" machines and attempts to playback their content on "N+1" machine, they will be asked to un-register one of their already registered machines before registering a new one. An un-registered machine removes the User's License disabling playback of User's content until the User chooses to re-license their machine.

2.2 Device Experience

This protocol defines a secure method of allowing a PC to securely provision a PD with a User Key. Once the PD has been provisioned with a user key or "registered" to that user, it instantly gains access to a users entire library of content. The content is simply directly transferred from the PC to the PD. The process of de-registering a device is simply the inverse of registration. The PC needs to verify that the PD is no longer capable of playing user's content.

3. Introduction

The Helix Device DRM protocol is designed to be easily adapted to PDs with varying characteristics. The PC and PD communicate via messages that can be passed asynchronously over any medium. This flexibility allows the messages to be passed to directly via device drivers or indirectly via drop boxes on dumb storage devices.

The Helix Device DRM protocol is broken into three stages:

a.) Obtain Device Information (PD->PC)
b.) Registration Messages (PD<-PC)
c.) Device Message Processing (PD->PC) (optional response)

The first two stages are the same across all devices. Device messaging generally has the same flow and behavior across all devices. But the actual implementation of this processing will vary depending on the device characteristics. See Appendix A for the custom ways devices may choose to process rights depending on the device type.

3.1 Obtain Device Information

The PD must provide the PD specific information to the PC in a standard format. The PC uses the PD information to create device specific confidential messages.

The method used to transfer the information to the PC is left to the specific device implementation. The device can push its information to a storage device for the PC to pick up from a predefined location or a device may choose to have the PC pull the information from the PD via a proprietary message/request.

Section 4 details the device information format.

3.2 Registration Messages

Once the PC has gained access to the PD's information the PC should have everything it needs to transfer a key to the device. The PC can then create a message to register or de-register a user on the device.

Section 5 details the message protocol and format.

3.3 Device Message Processing

Upon receiving a message from the PC the PD must process and/or respond to the message. The device information must also be updated for future messages.

User registration messages will cause the PD to decrypt key updates and store new user keys securely in non-volatile memory. De-registration messages will cause the device to remove any requested user keys and generate a registration response message.

Section 6 details the message processing and response protocol. Appendix A details the requirements for devices with various characteristics. Each device will need to provide support documentation detailing how they fulfill the device requirements.

4. Device Information

Information is transferred from the device to the PC in two parts. The dynamic device information is generated dynamically on the device. The device certificate is static data, signed by RealNetworks and included with the device code and keys.

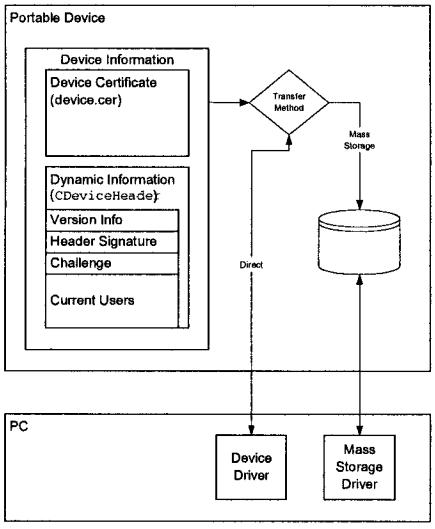


Figure 1 – Obtaining Device Information

4.1 Dynamic Device Information

The information should be generated on device re-set and updated after each message.

The `CDeviceHeader` structure defines the dynamic device information. All data is stored in network byte order.

`CDeviceHeader` is defined as:

```
typedef _CDeviceHeader
{
     // the magic number is defined to be 0x524e4442
     UINT32             magic_number;
     // size of this structure
     UINT16             header_size;
     // signed hash of the rest of this structure
     unsigned char      header_signature[128];

// begin signed information
     // version of this structure - currently: 0x0000
     UINT16             version;
     // unique device identifier
     unsigned char      device_id[20];
     // unique challenge generated by the device
     unsigned char      challenge[16];
     // the current users registered on this device
     CCurrentUsers      current_users[];
// end signed information
} CDeviceHeader;
```

CCurrentUsers is defined as:

```
typedef _CCurrentUsers
{
      // number of users in this structure
      UINT16      user_count;
      // a variable length list of GUIDs representing the registered
      // users on this device
      GUID        current_users[];
}  CCurrentUsers;
```

GUID is defined as:

```
typedef struct  _GUID
{
      ULONG32     Data1;
      UINT16      Data2;
      UINT16      Data3;
      UCHAR       Data4[ 8 ];
}GUID;
```

The `CDeviceHeader` structure is created on the PD. This should generally be created on the PD when the PD is started up. It also can be generated dynamically though. The following properties must be observed:

- The current users authorized to consume music on the device MUST be listed in the list of current users.
- The device ID is the unique device identifier. The unique device identifier is a constant and public identifier used to identify an individual device.
- The challenge is a unique number generated by the device. It should be initialized to be a random number must be changed after every block of messages is processed.

The signed_header is signed/encrypted using the RSA private key or shared symmetric secret hold by the device. Its internal structure is defined as:

```
Typedef _CChallenge
{
      unsigned char     db_hash[20];
      ULONG32           magic_number;
      unsigned char     randomdata[127-20-4];
} CChallenge;
```

The magic_number is defined to be 0xDEADBEEF.

challenge_data is a 16 byte binary data maintained by the device. It should be changed to a new value by the device for each dropoff DB read/write operation. Device should internally maintain a copy of the current challenge_data and always compare two values to verify that dropoff DB is a current one, instead of a previously seen one to prevent replay attack. see [TBD] section for the robustness requirement on challenge_data and possible implementation.

Device must initialize at least the dropoff DB header, before the communication between the device and PC can start. PC will not create a dropoff DB if one is not present.

4.2 Device Certificate

The first part is the device certificate. The device certificate is individualized for devices that use asymmetric cryptography. The devices public key is contained in the certificate for each device.

Devices using symmetric cryptography will also have a certificate. The certificate will contain the devices private key encrypted with a RealNetworks trusted key. The PC will decrypt this key in trusted tamper resistant code.

Each player device comes with a unique X.509 format certificate embedded. The certificate is in PEM format (base64 encoded DER) and is signed by RealNetworks. The signature algorithm is sha1WithRSAEncryption.

The certificate will be slightly different depending on whether the implementation is the asymmetric or symmetric protocol variant. The asymmetric certificate will contain device's 1024 bit RSA public key for encryption. The symmetric certificate will include a device class identifier, so the server can associate the certificate with the appropriate secret.

A player device should treat the certificate as opaque data. The private or secret key shall be guarded with appropriate key hiding and tamper resistance techniques (see robustness section 9.0). When a device is attached to a new piece of storage (in the case of removable media), the device should copy its certificate to a specific directory on the storage media.

The directory and file name for the certificate would be: \helix_d\device.cer.

Comment [U1]: Sheldon to update to reflect the fact we will use our own certificate format and the symmetric certificates will contain the asymmetrically encrypted user key.

4.3 Certificate Revocation

When the PC receives a certificate it first must validates the certificate and checks the certificate against its revocation list. It also checks the device ID against its revocation list. If the client is revoked the PC issues a message to de-register all the users on the device. TODO – add certificate revocation to figure 1

5. Registration/De-Registration Messages

After the PC has retrieved the device information it can create registration and de-registration messages. These messages can then be transferred to the device.

Both registration & de-registration have a structure. After the message is created on the PC it is appended to the `CDeviceHeader` structure and transferred back to the device. Thus a registration message would be the following structure:

```
typedef struct _CRegistrationMessage
{
     CDeviceHeader dynamic_device_header;
     CRegisterUser register_user[];
} CRegistrationMessage;
```

Note multiple users can be registered on a device at the same time. The messages must be linked to the same `CDeviceHeader` structure.

The `e_struct_type` enum is defined to differentiate the possible message types that can be attached to a `CDeviceHeader` structure:

```
typedef enum
```

```
{
      e_media_UserKey;
      e_media_DeRegisterDeviceRequest,
      e_media_DeRegisterDeviceResponse
} e_struct_type;
```

5.1 Registration Process

[TODO]

Comment [U2]: Section Needs updated/edited

When the PC is requested to authorize a device, it first retrieves the device information from the device.

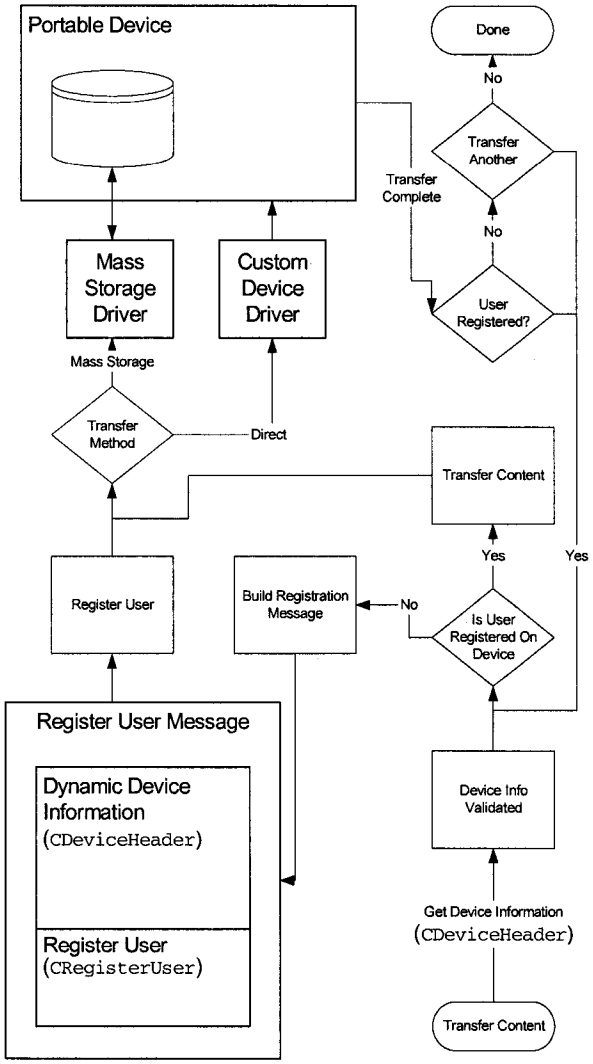


Figure 2 – Device Registration Process

If the certificate passes verification the PC checks the CDeviceHeader to see if the user to be registered is already on the PD. If the user is not registered on the PD the server checks to see if the user has rights to register a new PD. If the user has sufficient rights, the CRegisterUser structure is appended to the CDeviceHeader creating a registration message.

5.1.1 Registration Messages

The `CRegisterUser` structure has the following form:

```
typedef struct _CRegisterUser
{
      ULONG32           size;
      e_struct_type     type;
      GUID              userID;
      CUserKey          encryptedKey;
} CRegisterUser;
```

The `CUserKey` structure is encrypted with the key specified in the device certificate.

In the symmetric case an individualized transfer key is calculated. The key is an HMAC of the transfer key and the device id. The transfer key is determined by decrypted by the DRM using the RealNetworks private key (from with-in tamper resistant code).

The asymmetric protocol encrypts `CUserKey` with the RSA 1024 bit key in the device certificate.

In the case of the symmetric protocol, `CUserKey` is encrypted with the server's shared symmetric secrete. `CUserKey` is defined as follows:

```
typedef _CUserKey
{
      unsigned char           key[16];
      unsigned UINT32         magic_number;
      unsigned char           challenge[16];
      unsigned char           device_id[20];
      unsigned char           randomdata[128-4-16-16-20];
} CUserKey;
```

The magic_number is defined as 0x20204f4b. The challenge & device id are copied from the CDeviceHeader structure. The device will verify the magic_number, challenge, and device_id to validate the CUserKey record.

5.2 De-Registration Process

[TODO]

Comment [U3]: Section Needs updated/edited

When the user wishes to deregister a PD the PC software creates a CDeRegisterDeviceRequest message and sends it to the PD. In order to create a CDeRegisterDeviceRequest message the PC will need to create a new request ID and encrypt it similar to how it would encrypt a key registration message.

The user's count of authorized devices is NOT decremented on the PC software at this point. When the PC receives the new message it should process it accordingly.

Processing a CdeRegisterDeviceRequest will involve removing the key/user pair from its internal non-user accessible ram/flash, and then returning a CDeRegisterDeviceResponse to the PC. The response message is appended to an updated CDeviceHeader message and passed back to the PC.

The PC should wait for a new CDeRegisterDeviceResponse structure after it has issued a de-registration request. It will check then check the returned registration ID, assuming the id is valid, the server will decrement the user's count of authorized devices.

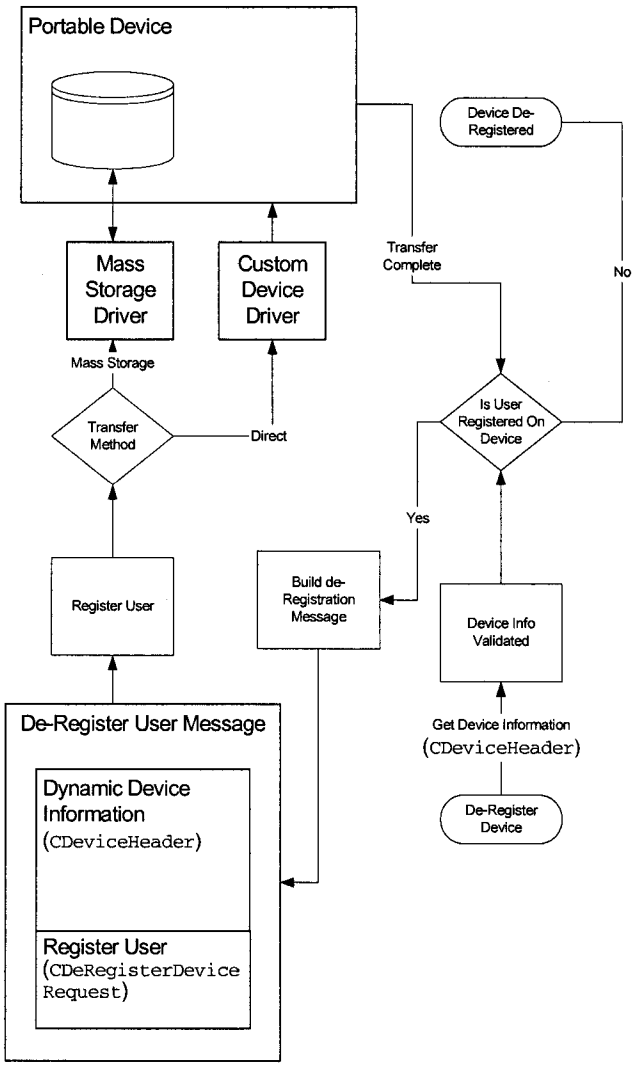


Figure 3 - Device De-Registration

5.2.1 De-Registration Messages

```
typedef struct _CDeRegisterDeviceRequest
{
      ULONG32           size;
      e_struct_type     type;
      GUID              userID [16];
      Unsigned char     request_id[16];
} CDeRegisterDeviceRequest;
```

5.2.2 De-Registration Response Message

```
typedef struct _CDeRegisterDeviceResponse
{
      ULONG32                 size;
      e_struct_type           type;
      GUID                    userID [16];
      CResponseSignature      encrypted_request_id;
} CDeRegisterDeviceResponse;
```

The `CResponseSignature` structure:

```
typedef _ CResponseSignature
{
      unsigned char     hash[20];   // a SHA-1 hash of
      challenge
      unsigned char     random_data[108];
} CResponseSignature;
```

The "responseSignature" contained within the CDeRegisterDeviceResponse is a RSA (using the key defined in section 4) signed CResponseSignature. The secret key is used instead in the case of the symmetric version of the protocol. CResponseSignature is defined as follows:

6. Device Message Processing

Upon receiving a new message from the PC the PD the client hardware, will parse the messages. If a CRegisterUser message is received, the device will decrypt the record, verify the device_id & challenge match the internally stored challenge and then proceed to store the userid's and keys within it's private storage area (non-user accessible non-volatile memory).

On attempt to playback content that is not authorized on the client, the client should check the drop-off point to see if a new user key has been given.

TODO write up device processing.

Comment [U4]: needs updated

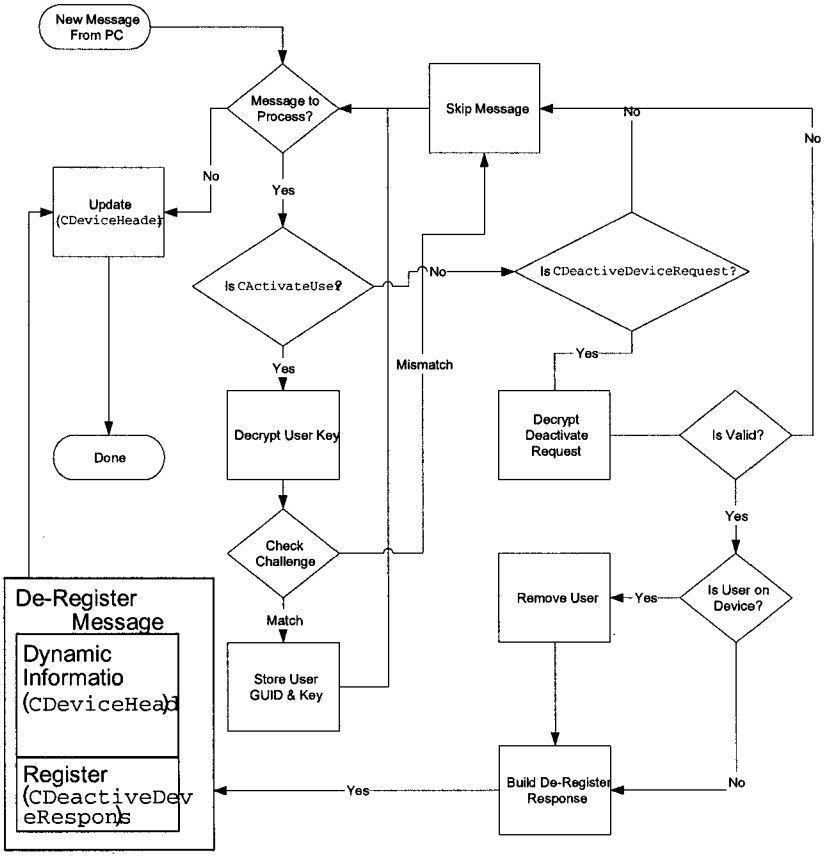


Figure 4 - Message Processing On Device

7. Appendix A – Requirements for on device message processing

TODO – write up different classes of device requirements:

Comment [U5]: Needs updated

Mass storage – internal "private" storage: (Nokia & Archos) (painful because we have to disconnect device to de-register)

Custom drivers, internal private storage: (Creative)

Custome drivers – No volatile storage: (Palm)

8. Appendix B – Compliance Requirements for Helix Device DRM Licensed Devices

8.1 Applicability.

These rules are applicable to Licensed Products that have implemented playback functionality of Helix DRM Content.

8.2 OBLIGATIONS REGARDING PERSISTENT STORAGE OF CONTENT

Licensed products shall be constructed such that all data is treated as all Helix DRM content may not, once decrypted, be stored except for the sole purpose of enabling the immediate consumption of content but which (a) does not persist materially after the content has been consumed and (b) is not stored in a way which permits copying or storing of such data for other purposes.

8.3 PERMITTED OUTPUTS

8.3.1 Generally.

As set forth in more detail below, a Licensed Product shall not pass Decrypted Helix DRM content, whether in digital or analog form, to an output except as permitted below.

8.3.2 Analog.

There are no prohibitions relating to analog audio outputs.

8.3.3 Digital.

Licensed Products shall not output Helix DRM Audio content in digital form.

9. Appendix C – Robustness Requirements for Helix Device DRM Licensed Devices

9.1 Overview

Participating clients shall be designed and manufactured so as to resist attempts to circumvent the functions of the specification as more specifically described below.

9.1.1 Defeating Functions and Features

Participating clients shall not include: (1) switches, buttons, jumpers or software equivalents, (2) traces that could be cut, or (3) features/functions (eg: service menus) which could be used to defeat any function of the specification.

The list of functions which can not be thusly compromised includes but is not limited to:

1. Protection of the decrypted content
2. Protection of the RealNetworks Global Key or the RealNetworks Signed Individualized Private Key.
3. Enforcement of any associated playback restriction information.

9.1.2 Keep Secrets

Participating Clients shall be designed and manufactured such that they shall resist attempts to discover or reveal device keys or secret intermediate calculated cryptographic values.

9.2 Robustness requirements of software implementation

Any portion of a participating client that utilizes, in software, the RealNetworks Global Key/RealNetworks Signed Individualized Private Key and/or the decrypted User Key should use any and all reasonable methods to frustrate efforts to obtain this key.

All software implementations must at a minimum include: (1) techniques of obfuscation to disguise, hamper and frustrate attempts to discover the approaches used to hide the use and value of the keys (2) self-checking of the integrity of its component parts and be designed to result in a failure to provide decryption in the event of unauthorized modification.

Further, they may include, but are not limited to: (1) code encryption (2) execution in ring zero or supervisor mode (3) embodiment in a secure physical implementation that cannot be reasonably read.

9.3 Robustness requirements of hardware implementation

Any hardware component that utilizes the RealNetworks Global Key/ RealNetworks Signed Individualized Private Key, or that relies upon hardware to satisfy these robustness rules shall: (1) embed RealNetworks Global Key/RealNetworks Signed Individualized Private Key and/or the decrypted User Key in silicon circuitry or firmware which cannot reasonably be read (2) be designed such that attempts to remove or replace hardware elements in a way that would compromise the RealNetworks Global Key/RealNetworks Signed Individualized Private Key and/or the decrypted User Key would pose a serious risk of damaging the Participating Device. By way of example, a component, which is soldered rather than socketed, may be appropriate for these means.

9.3.1 Hardware Data paths

Decrypted data shall not be present on any user accessible buses in useable form. For this purpose user accessible bus is defined as: (1) an internal analogue connector which is designed for end user upgrades or readily facilitates end user access (2) a data bus that is designed for end user upgrades or access, eg: PCMCIA, CardBus, PCI bus.

A user accessible bus does not include memory buses, CPU buses or similar portions of a devices internal architecture.

9.4 Required Levels of Robustness

The levels of robustness described in sections throughout this section shall be implemented so that it is reasonably certain that they cannot be defeated or circumvented using "Widely Available Tools" or "Specialized Tools". And that only with great difficulty they can be circumvented with "Professional Tools."

9.4.1 Widely Available Tools

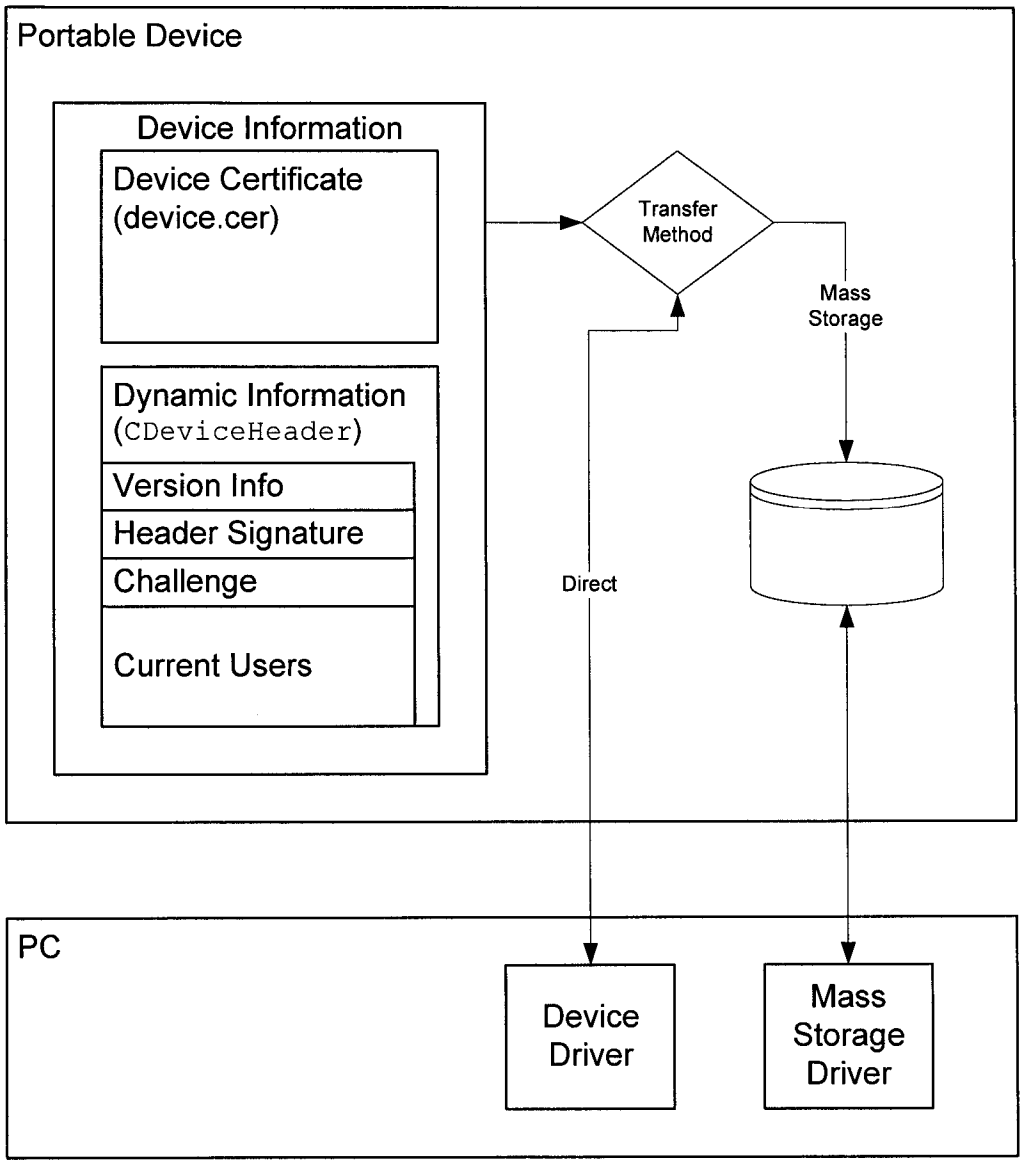
Widely Available Tools is defined as general-purpose tools or equipment that are widely available at a reasonable price, such as screwdrivers, jumpers, clips, file editors, and soldering irons.

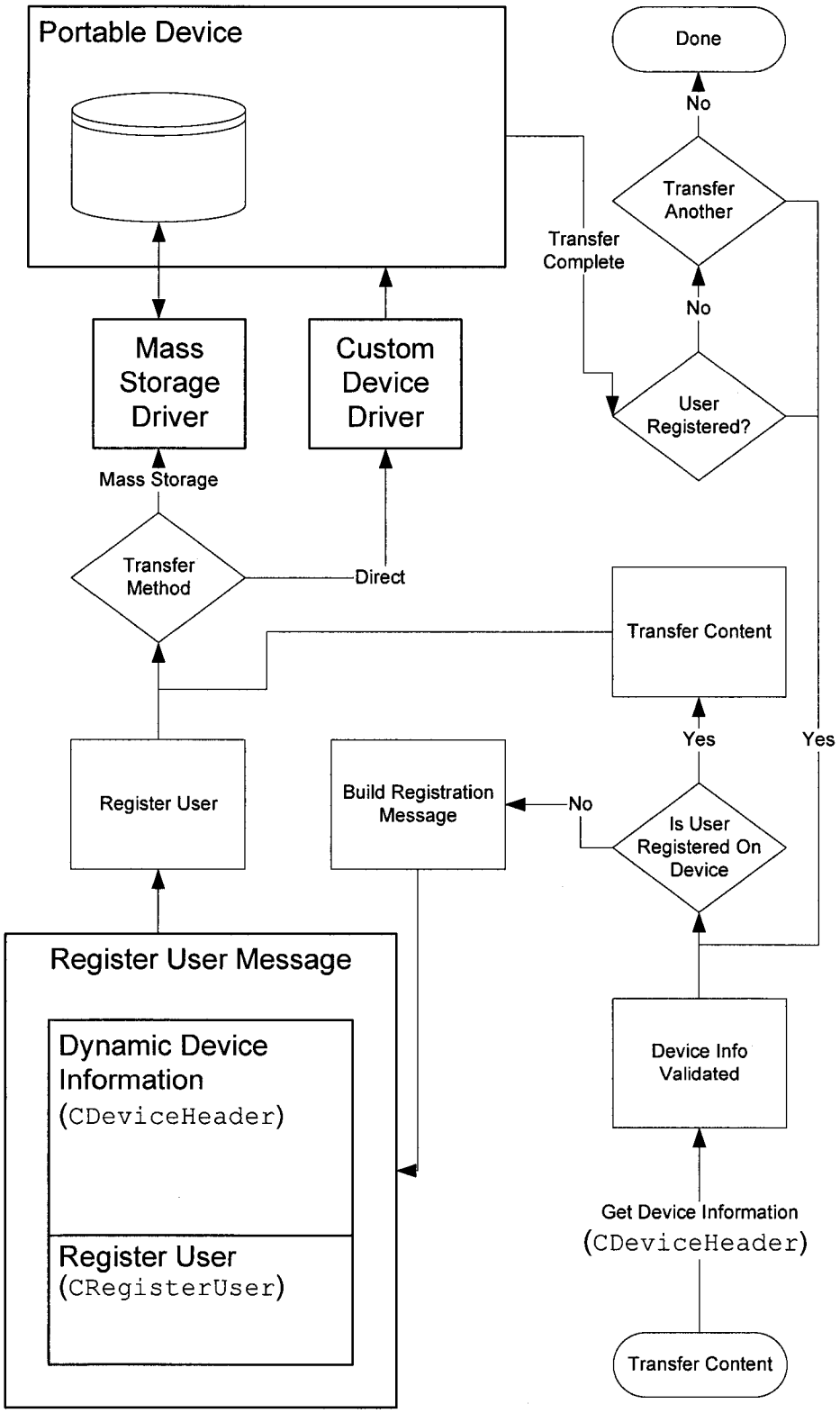
9.4.2 Specialized Tools

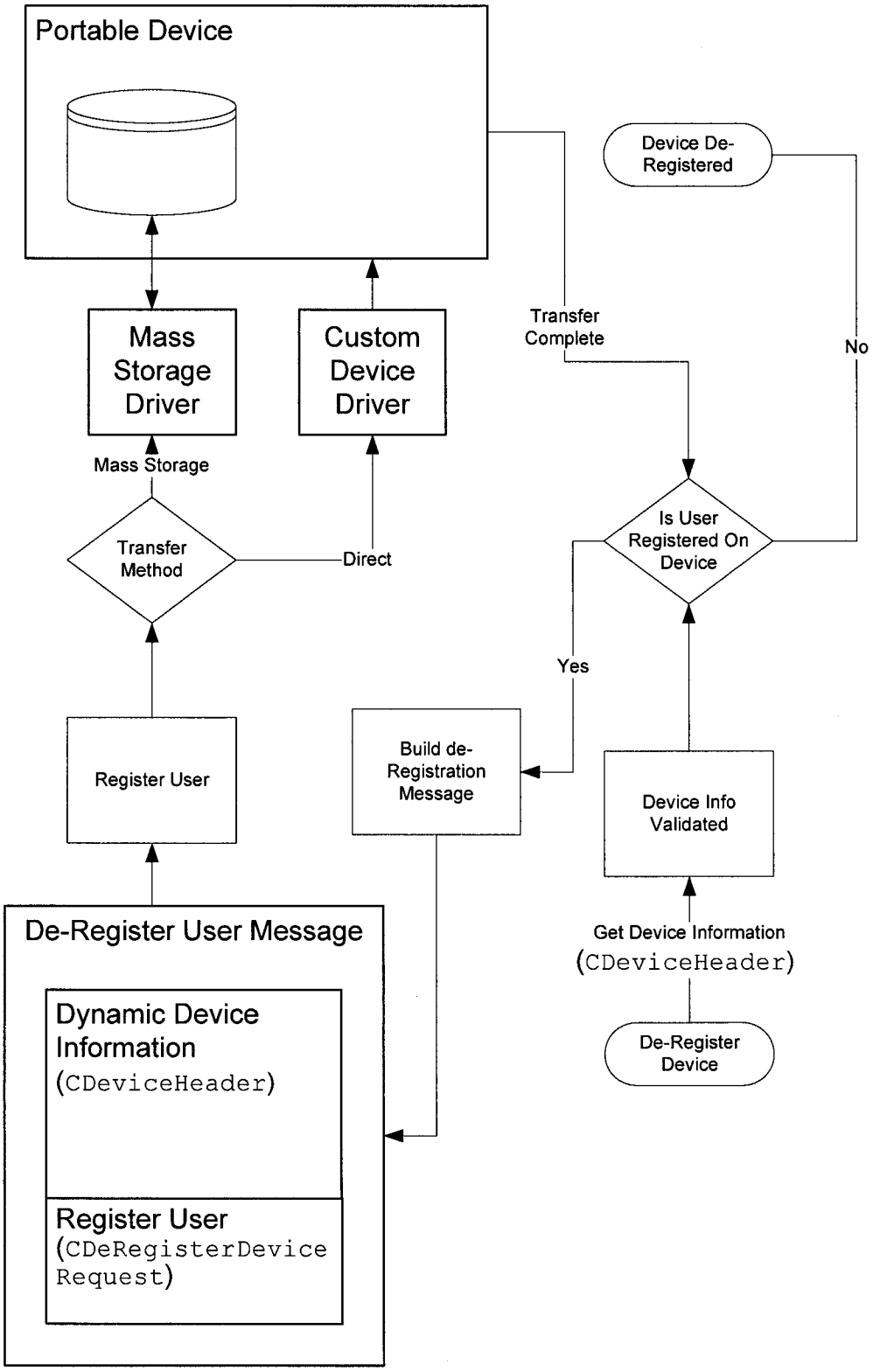
Specialized Tools is defined as specialized electronic tools that are widely available at a reasonable price, such as memory readers and writers, debuggers, decompilers, or similar software development products.

9.4.3 Professional Tools

Professional Tools is defined as professional tools or equipment, such as logic analyzers, chip disassembly systems, or in circuit emulators.


Portable Device
Device Information
Device Certificate
(device.cer)
Dynamic Information
(CDeviceHeader)
Version Info
Header Signature
Challenge
Current Users
Transfer
Method
Mass
Storage
Direct
PC
Device
Driver
Mass
Storage
Driver


Portable Device
Done
No
Transfer Another
Transfer Complete
No
User Registered?
Mass Storage Driver
Custom Device Driver
Mass Storage
Transfer Method
Direct
Transfer Content
Yes
Yes
Register User
Build Registration Message
No
Is User Registered On Device
Register User Message
Dynamic Device Information
(CDeviceHeader)
Register User
(CRegisterUser)
Device Info Validated
Get Device Information
(CDeviceHeader)
Transfer Content


Portable Device
Device De-Registered
Transfer Complete
No
Mass Storage Driver
Custom Device Driver
Mass Storage
Transfer Method
Direct
Is User Registered On Device
Yes
Register User
Build de-Registration Message
Device Info Validated
De-Register User Message
Get Device Information
(CDeviceHeader)
Dynamic Device Information
(CDeviceHeader)
De-Register Device
Register User
(CDeRegisterDevice Request)

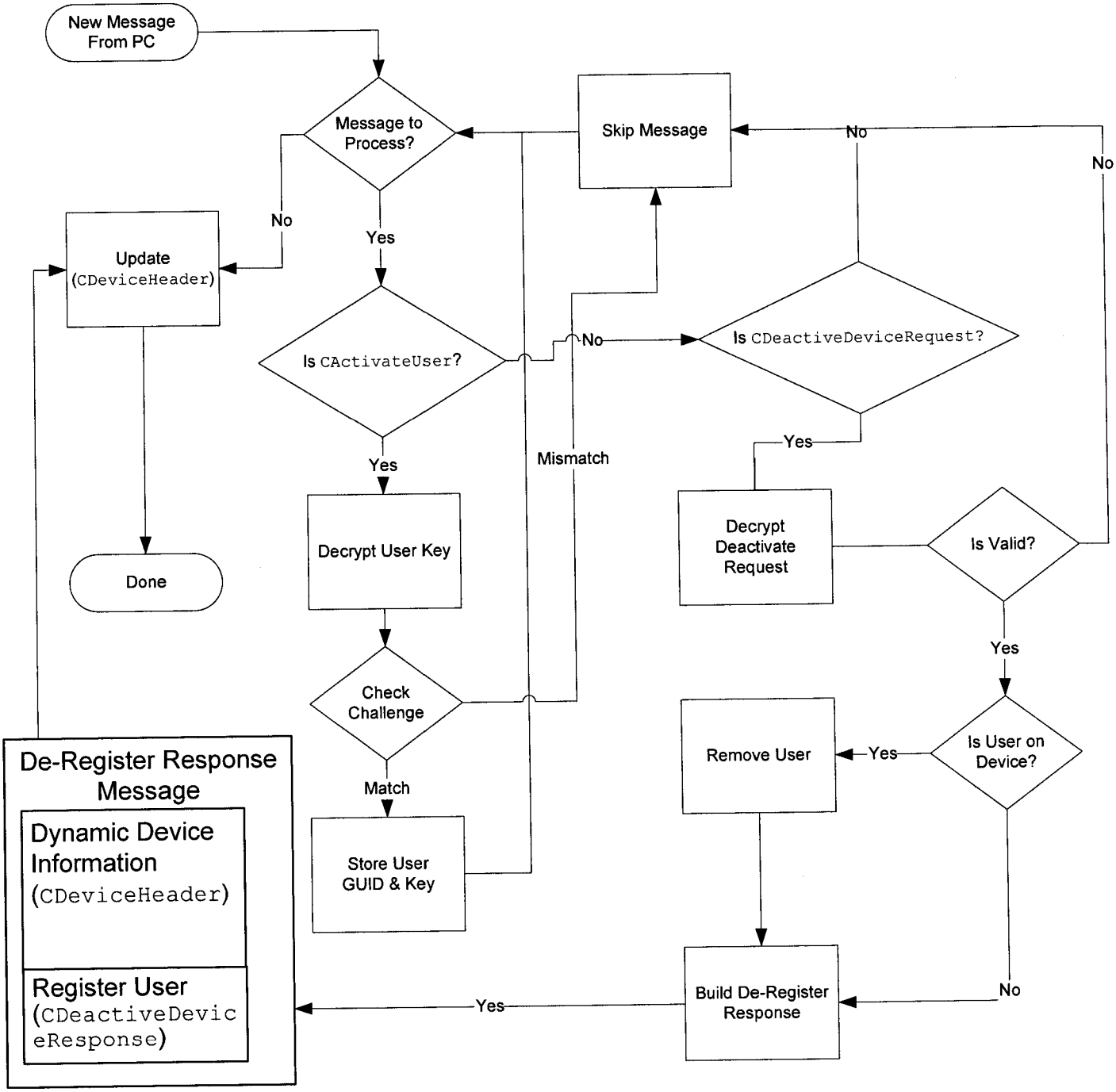

New Message From PC
Message to Process?
Skip Message
No
No
No
Update (CDeviceHeader)
Yes
Is CActivateUser?
No
Is CDeactiveDeviceRequest?
Yes
Mismatch
Yes
Decrypt User Key
Decrypt Deactivate Request
Is Valid?
Done
Yes
Check Challenge
Remove User
Yes
Is User on Device?
Match
Store User GUID & Key
Build De-Register Response
Yes
No
De-Register Response Message
Dynamic Device Information (CDeviceHeader)
Register User (CDeactiveDeviceResponse)

EXHIBIT H

Subject: Helix Device DRM.ppt
Date: Fri, 15 Aug 2003 16:29:23 -0700
From: Josh Hug <jhug@real.com>
To: Ranah Edelin <ranah@listen.com>, Rahul Agarwal <rahul@real.com>, Jeff Ayars <jeffa@real.com>, Adam Cappio <adamc@real.com>, Niranjan Nagar <niranjan@listen.com>, Dave Williams <dave@listen.com>, Ryc Brownrigg <rbrownrigg@real.com>, Evan Krasts <ekrasts@listen.com>

Here is my first draft of my presentations for the Music Labels to help further the Rights discussions.

Two presentations:
1.) The Music Experience w/ Helix DRM
Explain user licensing, and present issues w/ CD burning & PD transfers from a usability perspective.

2.) Helix Device DRM
Present our Device DRM. How it works, its security, and comparison to existing standards & technologies.

TODO:
Edit, and refine.

I plan to work on it a little bit tomorrow morning, and then I will have a long flight on Sunday. Monday morning I will meet w/ Ranah for last minute adjustments, re-organization & customizations for different meetings.

Thanks for any feedback and suggestions on organization and content.

Josh

Helix Device DRM Security Analysis

September 30, 2008

Objective

- Present Helix Device DRM Design
- Analysis of Helix Device DRM's security
- Comparison to other solutions

Design Goals

- Create familiar MP3 "like" user experience for honest users
- A user's license is transferred via the device initialization process
  - Access to this key provides access to a user's entire content library
- After the device is initialized user's content can be copied directly to the device
  - Media could be sent by the RealOne player or sent directly to the device or via alternative methods (removable media, mass storage device etc.)
  - The media format never changes
    - Same cryptographic protection and trust model

Protocol Details – Packaging

- Each file is individually encrypted with a content key (CEK)
- When a user purchases a track the track is individualized
  - Each user is assigned their own ID (UserGUID) and key (UserKey)
  - The CEK is then encrypted with the UserKey
  - This individualized blob is then inserted into the file's header (GUID, UserKey(CEK))

Protocol Details - Initialization

- Initialization of a device involves securely transferring a UserGUID and its associated UserKey
- Security
  - Use an individualized transfer key
  - Protocol protects against re-play attack
- Designed to require minimal interaction between the device and PC
  - Stateless based communication
  - Allows for easy implementation on devices with different characteristics
    - Pure mass storage devices
    - Devices with custom drivers
    - etc

Protocol Details – Playback

- The device then reads the UserGUID and UserKey(CEK) from the file header at playback time.
  - The UserKey associated with the UserGUID in the file is retrieved from the secure storage and used to decrypt the CEK
  - The CEK is then used to decrypt content before decoding and playback.

Security Analysis

- PC side of transfer protocol uses world class TR as Helix DRM
  - Includes: Individualization, obfuscation, tamper resistance, code execution in privileged mode.
- Clear compliance and robustness rules for device implementations
  - Ensured devices are secure and fit into the Helix Device DRM trust model
  - All Helix DRM devices must meet requirements in specifications

Compliance rules

- Clear compliance rules for implementations
  - More restrictive than CPRM or DTCP
    - No digital output allowed
  - Content can only be decrypted for immediate playback

Robustness rules

- Clear Robustness rules
  - Requirements are on par with SDMI & CPRM robustness requirements
  - All secrets must be protected by device hardware and software
- Software Rules & Hardware Rules robustness rules

Security level vs. Other Products

- Comparison to MS
  - User Licensing vs. Individual content licensing
  - Our user experience is better
  - Our security speaks for itself
- Comparison to Apple
  - Trust Model?

The Devices are coming

- Worked to get older versions of Helix DRM on devices
  - Were never satisfied with the user experience, thus we could not justify enough investment to bring to market.
- The usability benefits of our new version change things
  - We are actively working with multiple partners in multiple device categories to add support for our latest DRM to their devices.

Helix™ DRM

The Music Experience w/ Helix DRM

Minimizing the impact on User Experience

September 30, 2008

Objective

- Present our DRM Philosophy
- Identify the impact of different portability restrictions on usability
  - PC to PC portability model
  - CD Transfer portability
  - PD Transfer portability
- Present our recommend set of rights restrictions that will effectively protect media assets while allowing a good user experience

Philosophy

- DRM is about restricting usage
  - Our job is to protect media
  - Also enable sale of media to consumers
  - We have strived to make our restrictions seamless and invisible to users
- Replicate the usability of MP3's
  - DRM should be invisible to users who play by the rule's
  - Restrictions should only be felt when users attempt to do unauthorized actions
  - We don't want users to feel like they are in chains
    - We have learned lessons from past usability experience

Acceptable Restrictions

- Our challenge is to define restricted playback rules while still allowing for an acceptable user experience
  - We won't be successful asking users to pay for something that creates a worse user experience then what they are use to
  - To solve this, we can create software that replicates a familiar user experience
- When considering music purchase portability is the most important
  - PC to PC (our Helix DRM user licensing model)
  - PC to CD (our RealOne player Burn engine)
  - PC to PD (our Helix Device DRM specification)

The PC playback experience

- Helix DRM's is tested and approved
  - World class tamper resistance technology
  - Fully renewable via revocation
  - Has been analyzed and verified to meet bar to meet the security bar set by major movie studios and recording labels
- The Helix DRM supports three Licensing models
  - Individual Licensing
  - Subscription Licensing
  - User Licensing

Individual licensing

- Every piece of content is uniquely identifiable and can be licensed independently
- Benefits
  - Individualized control over each piece of media
- Challenges for music
  - Hard to manage large amount of similar content
    - What content has what rights? Confusing to users
  - Not a good experience for PD support
    - Have to copy a key w/ every file
    - Slow transfer rate, forced to use custom applications.
  - User experience is bad when content is moved to a second PC
    - Second PC has to be on-line
    - Must acquire each license individually
    - Hard and confusing when users want to change which computers are their authorized machines
  - Content expires individually and must be re-licensed individually

Subscription Licensing

- Individual licenses are grouped into "subscriptions"
- Access to the group of licensing is provisioned and managed by a single parent "subscription" license
- Benefits
  - Designed to allow for the dynamic update of a group of licenses on a recurring basis
  - As long as your subscription is current, you won't have to re-license your content.
- Challenges for Music
  - As in the individual licensing model Content still must be individually licensed creating a vary similar set of usability challenges
  - Solves the problem of licensing content on a recurring basis, but does not go far enough to improve portability and rights management

User Licensing

- Files are no-longer individually licensed and controlled
  - Same benefits as Subscription licensing, plus much simpler portability and rights management.
  - User license is protected with the same security and rights enforcement as all Helix DRM licenses
- Usability and portability benefits
  - One User License provides access to an entire body of content with similar rights
  - Simple for users to understand
    - Especially for music when users have hundreds of tracks
  - Effective protection and restrictions
    - Can express full set of rights restrictions on single User License – applied to all the content
  - Easy to register or de-register a machine
- Challenges
  - No control over individual tracks

User Licensing - PC User Experience

- User licensed content is "personalized" or "bound" to the Licensed User at the time of download.
  - The Content contains information about the User to whom the content is licensed, and it also contains the encrypted Content Key required to decrypt the content during playback.
  - The Content Provider assigns a unique User Key to each Licensed User.
  - The User Key is used to encrypt the Content Key in every User Licensed content file.
- To consume the User Licensed content file, the User must register their PC with the Content Provider.
  - Upon registration, the User will receive a License with the User Key and associated Rights.
  - The User License is kept in Secure Storage on the PC and is bound to the PC and to the User.
  - Access to the User License enables access to User's entire User Licensed content catalog.

User Licensing – PC to PC

- If a User wishes to use their content catalog on a different PC, they need to do two things:
  - Copy their User Licensed content to the new machine.
  - Register the new PC with the Content Provider, thus obtaining a License with the User Key.
- The user is restricted to register "N" machines.
  - If the User has already registered "N" machines and attempts to register their "N+1" machine, they will be asked to un-register one of their already registered machines before registering a new one.
  - An un-registered machine removes the User's License disabling playback of User's content until the User chooses to re-license their machine.

Music Portability

- Only allowing music playback on PCs limits usability
- Two portability methods are needed for digitally purchased music
  - Transfers to CD
    - CD players are everywhere and cheap
    - People currently expect to be able to play music they buy in CD players
  - Transfers to PD
    - PD's are becoming more popular with Users
    - Need to provide a secure method of transferring purchased music to these Devices.

Transfers to CD

- The Problem:
  - The restrictions we place on burning should not deter people from purchasing music
- The following usage restrictions have been proposed and we have investigated their impact on the user experience
  - Burn restrictions on individual tracks (i.e.)
    - 10 burns per track on one PC
    - 10 burns per track
    - 10 burns per track on each registered machine
  - Burn restrictions on play lists
    - 10 burns per play list
  - Combinations of the above

Usability Analysis – Burn Limits

- Generally user licensing was not designed to track individual counts associated with individual pieces of content
  - Requirement would eradicate many of the usability benefits associated with user licensing
  - Every piece of content would need an additional license to allow and count burning for each track
  - Significant amount of complexity is added when the track limits have been exceeded
- N burns on one PC
  - Clearly limits users ability to burn wherever they want
  - User must anticipate where they will do the majority of their burning – long term customer service problem:
    - What do we do when a user buys a new PC with better hardware?

Usability Analysis – Burn Limits

- N total burns for each track
  - Requires user to be connected during burn
  - Slows down burn process
  - Requires Users to authenticate with service whenever they burn
  - Requires an on line system to track the number of burns users have left on a track by track basis
  - A much more complex and slower user experience when compared with burning unprotected file
- N burns for each play list
  - No noticeable effect to the burn experience unless a play list is recorded N times
  - Effectively prevents people from pressing 100s of cd's for their friends.

Usability Analysis – Burn Limits

- General Issues with Burn Limits
  - Legitimate users should not feel limits
    - They are designed to be felt by people that break the rules
    - What is the difference between setting a limit of 10 and unlimited?
  - Confusing to customer since they just want to own the content and use it legitimately
  - Customer feels limited and confined, rather than free to use what they purchased

Transfers to PD

- The Problem:
  - Secure files can only be transferred to secure Digital Music Players. What are appropriate restrictions to be placed on track that a user has legitimately purchased.
- Helix DRM uses a registration model
  - Secure method is defined to allow a PC to securely provision a PD with a User Key.
  - Once the PD has been provisioned with a User Key, i.e. "registered" to the User, the device gains access to a User's entire library of content.
    - The content is simply directly transferred from the PC to the PD using any means available.
  - The process of de-registering a device is simply the inverse of registration.
    - The PC needs to verify that the PD is no longer capable of playing the User's content.

Familiar User Experience

- We want to provide familiar user experience to honest users
  - It should be silent and require no extra user interaction
- Restricting transfers sounds simple... But it quickly can grow into a very complicated user experience to express.
  - We are most concerned with keeping the user experience simple for legitimate users.
  - Multiple users can be registered on the same PC...
    - Therefore we believe we have to treat PD's as an extension of the PC's rights rather than an individual user's.
  - We have tried to diagram other methods and they become WAY to complicated.
    - Definitely not silent methods

Usability Analysis – Transfer Limits

- There are three restrictions that seem possible when we implement restrictions while considering the PD to be an extension of the PC's user set
  - Max Number of PD for each registered PC
  - When transferring a new user to a PD, the transferring PC must remove all users from the PD that are not currently registered on the transferring PC.
    - I.e. the PC must sync the users on the PD w/ the PC's set of users when it changes users on a PD.
  - Allow direct transfer to all secure and approved devices

Restriction 1 – Max Number of PD's

- We could envision a restriction on the number of devices a user could connect to a PC
  - Each PC would have a limited number of devices it could communicate with
  - When the users used up all their devices, they would have to un-register a device with the computer before they could register another.
- Challenges
  - While at first this sounds reasonable, it is not without issues.
    - Main issue is it requires user interaction for device registration

Restriction 1 – Max Number of PD's

- Challenges (cont)
  - We don't see most users having that many PD's...
    - We can't just register the PD's silently though...
    - They need to know that they are using one of the PC's limited number of slots.
  - On a multi-user PC, one user might use up all the allowed devices
    - Other users are left in the lurch.
  - No method for restoring device count for new PD's after one has been lost, broken or otherwise disposed of.
    - We don't have a secure method to allow our customer service to re-set their PC's device count.
    - Unless we created a back door in the security.

Restriction 2 – Sync Users

- Registering a user on a PD means synchronizing the PD's users w/ PC you are transferring from
  - This implies unlimited number of transfers but restricts devices ability to aggregate content from a number of different users.
  - If an old user is not on the new PC being transferred from, transferring a new user from the new PC will erase the old user from the device.
- Challenges
  - In some cases content would stop working on the device after syncing with a new PC.

Restriction 2 – Sync Users (cont)

- Users would run into this restriction in the following situation:
  - PC1 - UA & UB registered; PC2 - UB & UC registered
  - Register device to PC1... UA & UB are transferred
  - Go to PC2
  - UB content would transfer fine with no problems from PC2.
    - But when UC content is queued for transfer, we detect that the device will loose the rights to UA content if they proceed.
    - The user is then informed that activating UC content will mean UA content will no longer work on their device. (to remedy in help, they can either authorize UA on the new PC, or authorize UC on their old PC.)

Restriction 3 – Only secure devices

- Enabling transfer to all secure devices creates the best user experience
  - Fulfils the silent of silent registration
  - Users do not have that many devices
  - The content is still protected
  - We can track its usage when they try to play it on PC's
  - Devices have to connect to a registered PC to get a user's credentials

Recommendation

- User experience is the most important
  - The best user experience is achieved with
    - Max of 3 registered PC's
    - Unlimited burning
    - Unlimited transfers to Secure PD's

EXHIBIT I

Main Identity

From: "Sheldon Fu" <sfu@real.com>
To: "'Josh Hug'" <jhug@real.com>; "'Qiang Luo'" <qluo@real.com>; "'Adam Cappio'" <adamc@real.com>; "'Alain Hamel'" <ahamel@real.com>
Sent: Monday, February 09, 2004 4:24 PM
Attach: Helix Device DRM v.4.doc
Subject: groove revision 1

Was coding the implementation and found some problems =ith the usage reporting specified in initial draft. I added device_id and a new sequential number to the usage report message so that the =essage can not be faked from different device or subject to re-play attack. =o:p>

Even with that, there is still one problem left: =ince the spec calls for the device to reset the usage counters after generating a =sage report message, there is still a possibility of faking a PC request of =sage data and simply reset the counters in =esult.

We can not let the device keep the accumulated usage =ata, because over time, the internal log size will grow to infinity. We can =ot let the PC to detect skipped sequential number either because the device can =e hooked up to multiple PCs.

Only solution I can think about is to do a full =andshake and let PC send back a 'reset counter' message. But this =ill complicate both the PC and device side logic. =o:p>

What do you guys think? Do we really care so much =bout the authentication of the usage report data?

fxd

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

RealNetworks™

Helix Device DRM 2

Version 0.34

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Table of Contents

Deleted: 5
Deleted: 6
Deleted: 8
Deleted: 8
Deleted: 9
Deleted: 11
Deleted: 12
Deleted: 12
Deleted: 12
Deleted: 13
Deleted: 13
Deleted: 14
Deleted: 14
Deleted: 14
Deleted: 14
Deleted: 14
Deleted: 14
Deleted: 14
Deleted: 15
Deleted: 15
Deleted: 15
Deleted: 15
Deleted: 15
Deleted: 16
Deleted: 16
Deleted: 16
Deleted: 16
Deleted: 16
Deleted: 16
Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

1. Abstract

This document specifies the Helix Device DRM protocol. The protocol enables servers (PCs) to transfer protected media to trusted clients (devices). Devices are free to expose themselves to the OS in numerous ways. Two common methods are via a hierarchal file systems (ie. mass storage devices) and custom device drivers.

This protocol is a one-way destructive protocol. It is designed to allow the secure transfer of users from a PC to a device. When a device receives users, it MUST remove the current users before adding new users. Hence, it is called a destructive protocol.

2. Definitions

Content Key	Symmetric key used to encrypt a media file
Embedded License	Content Key encrypted with a User Key. The embedded License must also include the User ID.
Individually Licensed Content	Content requires an external license to manage its consumption.
Licensed User	User to whom a particular media file has been licensed
User	A back end service account, consisting of, at a minimum, an User ID and User Key
User ID	A 128 bit identifier linked to a Licensed User
User Inaccessible Memory	A storage area on a device designed to store keys and other information. These areas should only accessible by software running on the Device and not readily readable and accessible from the PC.
User Key	Symmetric key linked to a User ID used to encrypt Content Keys
User Licensed Content	Content including an Embedded License and associated User ID linking content with a particular Licensed User. This content is bound to this user regardless of the device consuming that content.

3. Background

The Helix Device DRM protocol is designed to authorize devices to consume User Licensed content. The following introduction explains the associated user experience when consuming User Licensed content on a computer and portable device.

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

3.1 User Licensed PC user experience

User licensed content is "personalized" or "bound" to the Licensed User at the time of download. The Content contains information about the User to whom the content is licensed, and it also contains the encrypted Content Key required to decrypt the content during playback. The Content Provider assigns a unique User Key to each Licensed User. The User Key is used to encrypt the Content Key in every User Licensed content file. To consume the User Licensed content file, the User must register their PC with the Content Provider. Upon registration, the User will receive a License with the User Key and associated Rights. The User License is kept in Secure Storage on the PC and is bound to the PC and to the User. Access to the User License enables access to User's entire User Licensed content catalog. Optionally, the user registration action could have a playback time-out value attached to it. The activated user key on the device will expire after the specified time-out amount of audio data has been consumed by the device.

If the User wishes to use their content catalog on a different PC, they need to do two things: 1) copy their User Licensed content to the new machine, and 2) register the new PC with the Content Provider, thus obtaining a License with the User Key. The Content Provider may allow the User to play back their content on up-to "N" registered machines. If the User has already registered "N" machines and attempts to playback their content on "N+1" machine, they will be asked to un-register one of their already registered machines before registering a new one. An un-registered machine removes the User's License disabling playback of User's content until the User chooses to re-license their machine.

3.2 Device User Experience

This protocol defines a secure method of allowing a PC to securely provision a device with a User Key. Once the device has been provisioned with a User Key, i.e. "registered" to the User, the device gains access to a User's entire library of content. The content can then be directly transferred from the PC to the device using any means available.

4. Introduction

The Helix Device DRM protocol is designed to be easily adapted to devices with varying characteristics. The PC and device communicate via messages that can be passed asynchronously over any medium. This flexibility allows the messages to be passed back and forth directly via device drivers or indirectly via drop boxes on dumb storage devices.

The Helix Device DRM protocol is broken into two stages:

a.) Obtain Device Information (Device→PC)
b.) Registration Message (Device←PC)

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

The required behavior during these stages is generally the same across all devices.

Additionally, there could be a usage report stage, during which the device reports back to the PC the statistical data about the consumption of the content on the device.

See "Appendix A – Requirements for message processing" to see the device specific requirements. Devices are required to provide documentation explaining how they fulfill these requirements.

4.1 Obtain Device Information

The device must provide the device specific information to the PC in a standard format. The PC uses the device information to create individualized and confidential messages.

The method used to transfer the information to the PC is left to the specific device implementation. The device can push its information to a storage device for the PC to pick up from a predefined location, or a device may choose to have the PC pull the information from the device via a proprietary request/response protocol.

Section 5, "Device Information", details the format of the messages.

4.2 Registration Message

Once the PC has gained access to the device's information, the PC should have everything it needs to transfer a key to the device. The PC can create a message to register its users on the device.

Upon receiving a message from the PC, the device must process and/or respond to the message. The device information must also be updated for future messages.

Section 6, "Registration Message", details the message protocol and format.

Formatted: Bullets and Numbering

4.3 Registration with Timeout Message

This is an extension of the 4.2 Registration Message. In additional to transferring the user key to the device, a timeout value is also transferred. The device must observe the timeout value and expire the user account after the specified amount of time has been consumed to play content under the user account.

Section 6, "Registration Message", details the message protocol and format.

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004
Formatted: Bullets and Numbering

4.4 Usage Reporting Message

Device sends this message at the request of the PC to report back the content usage information and purchase action conducted by the user on the device. Number of plays for each content since last usage reporting is transmitted back to the PC for auditing purpose. A flag is also transmitted for each contents that user marks as 'intent to purchase' on the device.

Details of usage reporting message is given in Section 7.

5. Device Information

Dynamic and static Information must be transferred from the device to the PC. The dynamic information is generated on the device; it must be generated for each request. The static data is contained in and referred to as the device certificate. The certificate is signed by RealNetworks and associated with highly confidential information embedded in trusted device code.

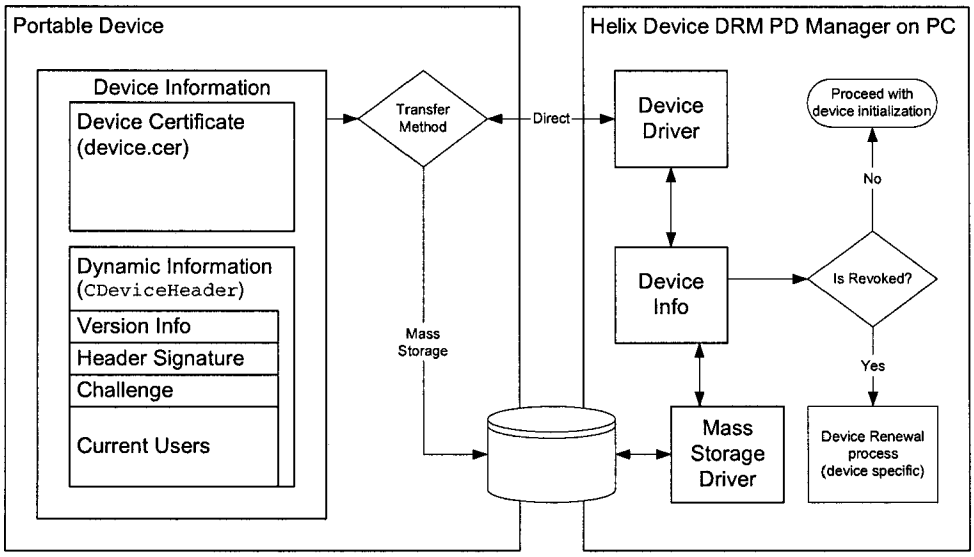


Figure 1 Device Information transfer and revocation

5.1 Individualization

The protocol can be implemented with asymmetric or symmetric cryptography. The asymmetric version of the protocol is designed for devices that have an individualized private key. The downside of the asymmetric method is individualization can be a burden on manufacturing. The symmetric variant lessens this burden while still maintaining an acceptable level of security.

Both methods provide for device revocation, and use an individualized transfer key. The difference lies in where the individualization occurs. The symmetric

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

key variant does individualization at key transfer time, whereas the asymmetric version individualizes the key at manufacturing time. In general, it is preferred to have individualization done at manufacturing time, but the costs associated can currently be prohibitive. Therefore, an alternative protocol allowing for dynamic individualization is provided.

5.2 Dynamic Device Information

The dynamic device information must be generated on the device and updated after each message.

The `CDeviceHeader` structure defines the dynamic device information. All data is stored in network byte order.

`CDeviceHeader` is defined as:

```
typedef _CDeviceHeader
{
     // the magic number is defined to be 0x524e4442
     UINT32              magic_number;
     // size of this structure
     UINT16              header_size;
     // signed hash of the rest of this structure
     unsigned char       header_signature[128];

// begin signed information
     // version of this structure - currently: 0x0000
     UINT16              version;
     // unique device identifier
     unsigned char       device_id[16];
     // unique challenge generated by the device
     unsigned char       challenge[16];
     // the current users registered on this device
     CCurrentUsers       current_users[];
// end signed information
} CDeviceHeader;
```

CCurrentUsers is defined as:

```
typedef _CCurrentUsers
{
      // number of users in this structure
      UINT16      user_count;
      // a variable length list of GUIDs representing the
      // registered users on this device
      GUID        current_users[];
}  CCurrentUsers;
```

GUID is defined as:

```
typedef struct  _GUID
{
      ULONG32     Data1;
```

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

```
        UINT16      Data2;
        UINT16      Data3;
        UCHAR       Data4[ 8 ];
    }GUID;
```

The `CDeviceHeader` is intended to be public data for the PD to advertise to the PC for pickup. Devices may choose to push the data to a mass storage device for the PC to read, or a device may generate the device info/and a new challenge dynamically for each request from a PC.

The challenge_data structure is a 16-byte binary integer maintained by the device. It should be changed to a new value after each message from the PC is processed. The device MUST internally maintain a copy of the current challenge in user inaccessible memory. The challenge maintained in user inaccessible memory is used to prevent replay attack. See Section 7.2, "Challenge", for the requirements placed on the challenge_data and possible implementation.

Deleted: 7.2
Deleted: 7.2

The `CDeviceHeader` structure is created on the device and must be created before communication between the device and PC can proceed. The following requirements are on the device info:

- The current users authorized to consume music on the device MUST be listed in the list of current users.
- The device ID is the unique device identifier. The unique device identifier is a constant and public identifier used to identify an individual device.
- The challenge is a unique number generated by the device. It should be initialized to be a random number, and it must not repeat to the same ID.
- A new signature with a fresh challenge should generally be created when the device is booted up. An exception would be when user inaccessible non-volatile memory is available to store the signature.
- The signature is signed/encrypted using the RSA private key or the device symmetric secret held by the device.
- The device should routinely verify the challenge_data in the current device info is the same as what is stored in non-user accessible memory

5.3 Device Certificate

The static device information is contained in the device certificate.

RealNetworks provides a unique certificate embedded to embed in each player implementation. The certificate's format will be slightly different depending on whether the implementation is the asymmetric or symmetric protocol variant. The asymmetric certificate will contain the device's 1024 bit RSA public key for encryption. The symmetric certificates will contain the device's private symmetric key encrypted with a public RealNetworks RSA trusted key. The PC will decrypt this key using the RealNetworks trusted private key that is protected by PC tamper resistant code.

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

The certificate is base64 encoded and generally should be treated as opaque data. The corresponding private or secret key MUST not be stored unprotected in user accessible memory. It must always be guarded with appropriate key hiding and tamper resistance techniques.

5.4 Certificate Revocation

When the PC receives a certificate, it first must validate the certificate and check the certificate against its revocation list. It also checks the device ID against its revocation list. If the client is revoked, the PC issues a message to de-register all the users on the device.

6. Registration Message

After the PC has retrieved the device information, it can create a registration message. This message is transferred to the device to register the users on the PC with that device.

After the registration message is created on the PC, it is appended to the `CDeviceHeader` structure and transferred back to the device. Thus, a registration message has the following structure:

```
typedef struct _CRegistrationMessage
{
     CDeviceHeader  dynamic_device_header;
     CRegisterUsers register_users;
} CRegistrationMessage;

or

typedef struct _CRegistrationWithTimeoutMessage
{
     CDeviceHeader  dynamic_device_header;
     CRegisterUsersWithTimeout register_users;
} CRegistrationWithTimeoutMessage;
```

6.1 Registration Process

When the PC wants to transfer content to a device it first retrieves the device information from the device.

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

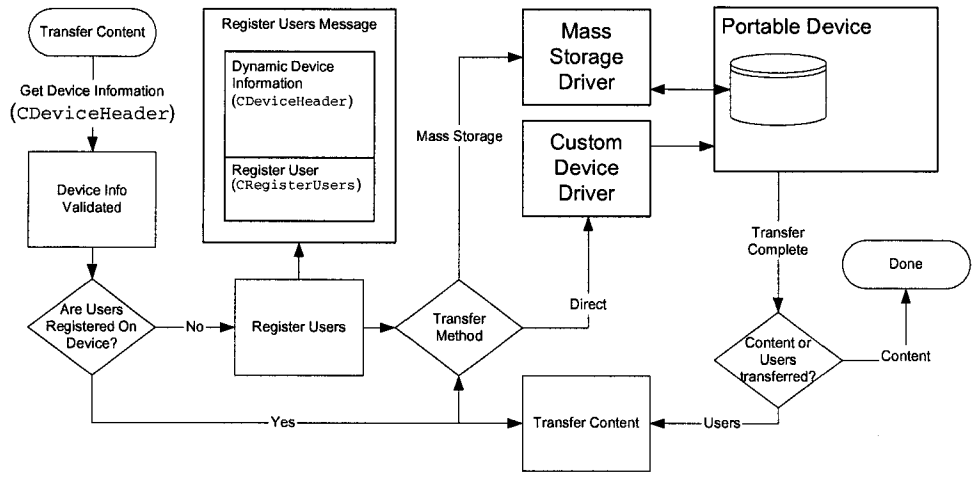


Figure 2 – Device Registration Process

After verifying the certificate is valid, the PC checks the `CDeviceHeader` to see if the user associated with the content to be transfer is already on the device. If the user is on the device, the PC can transfer the content freely without updating the device.

If the user is not currently on the device, the PC checks to see if the user has rights to transfer to this device. If the user has sufficient rights, the `CRegisterUsers` structure is created and appended to the `CDeviceHeader` creating a registration message.

6.2 Registration Messages

The `CRegisterUsers` structure has the following form:

```
typedef struct _CRegisterUsers
{
      UINT32            size;
      UINT16            num_users;
      CTransferKey      transferKey;
      CUserKey          user_keys[];
} CRegisterUsers;
```

`CTransferKey` is defined as follows:

```
typedef _CTransferKey
{
      unsigned char           transfer_key[16];
      unsigned char           hash[16];
      unsigned char           randomdata[128-16-16];
} CTransferKey;
```

The hash field in `CTransferKey` is a SHA1 hash of the following structure:

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

```
typedef struct _CTransferKeyHash
{
      unsigned char     transfer_key[16];
      unsigned char     challenge[16];
      unsigned char     device_id[16];
} CTransferKeyHash;
```

The challenge & device id are from the `CDeviceHeader` structure. The device will re-create the `CTransferKey` to verify the challenge and `device_id` from the PC thus validating the `CTransferKey` record.

The `CTransferKey` structure is encrypted with a device key from the device certificate.

In the symmetric case, an individualized device key is used to encrypt the `CTransferKey` structure. The encrypted device key is extracted from the certificate and decrypted using the RealNetworks private key from within tamper resistant code. An HMAC is created with the device key and the device's unique id, thus individualizing the device key.

The `transfer_key` is a 16-byte key generated by the device. This key is used to encrypt each individual User Key. Note PC's using should use a robust random number generator meeting the requirements required for a Helix DRM random number generator.

The asymmetric protocol encrypts `CTransferKey` with an RSA 1024 bit key in the device certificate.

```
typedef _CUserKey
{
      GUID              user_guid;
      unsigned char     key[16];
}  CUserKey
```

The User Key is stored in `key` and symmetrically encrypted with `transfer_key` using AES.

Formatted: Bullets and Numbering

6.3 Registration With Timeout Messages

The `CRegisterUsersWithTimeout` structure has the following form:

```
typedef struct _CRegisterUsersWithTimeout
{
      UINT32              size;
      UINT16              num_users;
      CTransferKey        transferKey;
      CUserKeyWithTimeout      user_keys[];
} CRegisterUsersWithTimeout;
```

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

The CTransferKey is defined in identical to the one in the 6.2 Registration Messages. The CuserKeyWithTimeout is defined as:

```
typedef _CUserKeyWithTimeout
{
      GUID              user_guid;
      unsigned char     key[16];
      UINT32            timeout;
      UINT16            account_type;
} CUserKeyWithTimeout;

#define AT_PERMANENT     0   //reserved for the old message
#define AT_SUBSCRIPTION  1
#define AT_PREVIEW       2
```

Deleted:

The User Key is stored in `key` and symmetrically encrypted using AES algorithm. The encryption key is formed by XORing `transfer_key` with timeout value in a byte by byte fashion for 4 bytes, starting with the offset zero in key[] array and the most significant byte of timeout

The unit of timeout value is in second. timeout value zero specially means infinite timeout. For a non-zero timeout value, the device should make the user key record expires after it has played content under the specific user account for the timeout amount of time.

Only three account types are defined. A permanent account is activiated using the CRegisterUsers message or using the CRegisterUsersWithTimeout message but with the timeout value equals zero. Both subscription and preview accounts have a non-zero timeout value associated with them.

Since there is no header information to distinguish a Registration Message from a Registration With Timeout Message, It is up to the link level implementation of the protocol to signal the device to process the incoming message as Registration Message or Registration With Timeout Message.

Formatted: Bullets and Numbering

6.4 Device Message Processing

Upon receiving a new registration message from the PC, the device will parse the message. The device will first decrypt the message and re-generate the `CTransferKeyHash` to verify the device_id & challenge match the device id and internally stored challenge.

Next the device SHALL delete all reference to any users that might have been registered via prior messages. The device will then proceed to store the new User IDs and User Keys in a user inaccessible storage area.

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

On attempt to playback content that is not authorized on the client, the client should check for a new message to verify a new message has not been provided.

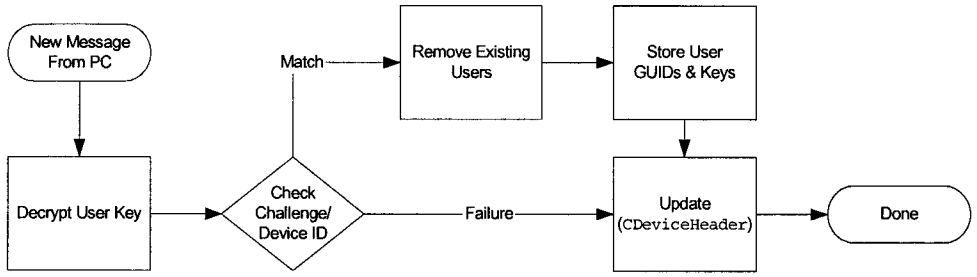


Figure 3 - Message Processing On Device

Formatted: Bullets and Numbering

7. Usage Reporting Message

Formatted: Bullets and Numbering

This message from the device reports back to the PC the number of plays and user purchase action for contents since the previous report message. Contents not played or not marked as 'intent to purchase' are not reported.

```
typedef struct __CUsageReport
{
        // the magic number is defined to be 0x524e4442
        UINT32              magic_number;
        // size of this structure
        UINT16              message_size;
        // signed hash of the rest of this structure
        unsigned char       signature[128];

        // begin signed information
        // version of this structure - currently: 0x0000
        UINT16              version;
        // unique device  identifier
        unsigned char       device_id[16];
        // sequential number
        UINT32              seq;
        // nmber of user records
        UINT16              num_users;
        CUserUsage          user_usage[];
// end signed information
} CUsageReport;

typedef struct __CUserUsage
{
        GUID                user_id;
        UINT16              num_items;
        CUsageRecord        usage[];
} CUserUsage;
```

Deleted: unique device identifier
Deleted: unsigned char device_id[16];

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

```
typedef struct _CUsageRecord
{
      GUID                content_id;
      UINT16              play_count;
      UINT16              flag;
} CUsageRecord;

#define FLAG_PURCHASE 0x0001
```

Other flag bits are reserved and can be used in the future.

'seq' is the usage report sequential number. Device should increase this number by one each time it generates a usage report message and reset the internal usage counters.

Content play count and flag are reset to zero after each Usage Report message is generated. It is up to the PC DRM to maintain a accumulated count if so desired.

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Helix Device DRM compliant device needs to appropriately implement several protocol related features for security reasons.

7.1 Device ID

Device ID in the message header is used by PC DRM to uniquely track each individual device. The device implementation should have a device ID that is not modifiable by the end user. A per device unique ID pre-embedded in the device ROM is the best approach. The Device ID must be based upon at least one unique device hardware identifiers. If the Device ID is based on a combination of identifiers, the Device ID can be generated at an appropriate time (such as the first time device DRM runs) and saved to some non-volatile, non-user-accessible storage locations. Writable flash and a secret area on mass storage are two appropriate examples.

7.2 Challenge

The challenge value in the message header is used for replay attack prevention. It should be changed each time the device processes a new message in the body.

For a device architecture that has a built-in, efficient random number generator, the challenge value should be generated as a random number when it needs to be changed. Some modern MCU/DSP architectures have a RNG built-in.

For device architecture that has an RNG, but the RNG is slow or expensive, an initial random challenge number should be generated at the device reset/startup or whenever the message header is lost. In this case, the challenge value can be simply increased by one when it is needs to be changed. Note that the protocol doesn't depending on the un-predictability of the next challenge value, so simply increasing the value by one has the same effect as using a pseudo RNG to modify the challenge value.

Whenever possible a RNG mechanism should be utilized as to provide the best protection against replaying messages. One option is to continually collect and store randomness as the user interacts with the device, sampling key presses or other input.

Device without RNG capability will suffer from a replay attack at device reset. For infinite licenses, replaying device keys is not a security issue. Lack of support for generating random numbers should be noted in the device certificate. Devices without a RNG will not be allowed to process user keys in the future that might contain timeouts our keys that need to be refreshed.

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

7.3 Protection for the message encryption key

In both symmetric and asymmetric version, the device needs to protect a secret device key (AES key or RSA private key). Adequate effort should be provided to hide these secret keys. Key hiding algorithm that blends the key into algorithm code is suggested to achieve this requirement. In no condition should these secrets keys be backed up in an unencrypted form to user-accessible storage or PC. The key should be held as highly confidential information and the robustness rules in "Appendix C – Robustness Requirements" should be followed.

7.4 Device internal key database

The device needs to maintain an internal database for user_id/user_key lookup. This internal database needs to be adequately protected since it contains user keys. Encryption is required for the database records. Since a User Key can be revoked from the device, the device needs to provide adequate protection against rollback attack. One example of rollback attack prevention is to calculate a database hash and save the hash in a separate user inaccessible place. The key database should not be transferable to other media for backup purposes, even if the device offers the general backup feature for other data on the device.

For devices with internal user-inaccessible and writable storage area, the key database should be saved to such a location. This storage area should be user inaccessible.

Device without internal permanent writable storage area such as portable player based on external memory card should keep the key database in its internal memory. Such a device will lose its user keys and need to be restored from the PC when the device is reset or looses power.

Formatted: Bullets and Numbering

7.5 Device Certificate generation and handling

The reference code & documentation will initially come with a development keys and certificates. These keys should be used to create the device implementation and test it to insure that it follows the specification.

Each device implementation should embed a unique key. When production keys are needed the implementer will need to request keys from us. All communication of device keys and associated certificates will be sent to a single individual utilizing PGP encryption. These keys are to be treated as highly confidential information.

[Insert rules for handling keys and certificate from DRM license agreement]

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004
Formatted: Bullets and Numbering
Formatted: Normal

7.6 Implementation of Account Timeout and Usage Counting

Account timeout is based upon actual number of seconds with which contents are played under the account, not the wall clock time. Timeout counting usually can be conducted in two places:

- After the decoder, before the PCM data is sent to audio device. Number of samples sent to audio device can be converted to be in unit of seconds and added to the timeout counter.

- When packets are decrypted by the DRM. Knowing the packet length in seconds, the timeout counter can be updated accordingly.

Usage counting should also be based on the actual content consumption. Two rules need to be considered:

Formatted: Normal

- Threshold for increasing the counter. Only after a content has been played for more than 30 seconds or half of its length if the total length is less than 30 seconds that the usage should be debted.
- Threshold for repeating plays. If the content has been played for more than 1.5 times of its length, it is considered another play.

Deleted: ¶

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

8. Appendix B – Compliance Requirements

8.1 Applicability.

These rules are applicable to Licensed Products that have implemented playback functionality of Helix DRM Content.

8.2 Obligations regarding the persistent storage of content

Licensed products shall be constructed such that all Helix DRM Content is treated with care after decryption. Helix DRM content may not, once decrypted, be stored except for the sole purpose of enabling the immediate consumption of content but which (a) does not persist materially after the content has been consumed and (b) is not stored in a way which permits copying or storing of such data for other purposes.

8.3 Permitted outputs

8.3.1 Generally.

As set forth in more detail below, a Licensed Product shall not pass Decrypted Helix DRM content, whether in digital or analog form, to an output except as permitted below.

8.3.2 Analog.

There are no prohibitions relating to analog audio outputs.

8.3.3 Digital.

Licensed Products shall not output Helix DRM Audio content in digital form.

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

9. Appendix C – Robustness Requirements

9.1 Overview

Participating clients shall be designed and manufactured so as to resist attempts to circumvent the functions of the specification as more specifically described below.

9.1.1 Defeating Functions and Features

Participating clients shall not include: (1) switches, buttons, jumpers or software equivalents, (2) traces that could be cut, or (3) features/functions (eg: service menus) which could be used to defeat any function of the specification.

The list of functions which cannot be thusly compromised includes but is not limited to:

1. Protection of the decrypted content
2. Protection of the RealNetworks Global Key or the RealNetworks Signed Individualized Private Key.
3. Enforcement of any associated playback restriction information.

9.1.2 Secrets Protection

Participating Clients shall be designed and manufactured such that they shall resist attempts to discover or reveal device keys or secret intermediate calculated cryptographic values.

9.2 Robustness requirements of software implementation

Any portion of a participating client that utilizes, in software, the RealNetworks Global Key/RealNetworks Signed Individualized Private Key and/or the decrypted User Key should use any and all reasonable methods to frustrate efforts to obtain this key.

All software implementations must at a minimum include: (1) techniques of obfuscation to disguise, hamper and frustrate attempts to discover the approaches used to hide the use and value of the keys (2) self-checking of the integrity of its component parts and be designed to result in a failure to provide decryption in the event of unauthorized modification.

Further, they may include, but are not limited to: (1) code encryption (2) execution in kernel or supervisor mode (3) embodiment in a secure physical implementation that cannot be reasonably read.

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004

9.3 Robustness requirements of hardware implementation

Any hardware component that utilizes the RealNetworks Global Key/ RealNetworks Signed Individualized Private Key, or that relies upon hardware to satisfy these robustness rules shall: (1) embed RealNetworks Global Key/RealNetworks Signed Individualized Private Key and/or the decrypted User Key in silicon circuitry or firmware which cannot reasonably be read (2) be designed such that attempts to remove or replace hardware elements in a way that would compromise the RealNetworks Global Key/RealNetworks Signed Individualized Private Key and/or the decrypted User Key would pose a serious risk of damaging the Participating Device. By way of example, a component, which is soldered rather than socketed, may be appropriate for these means.

9.3.1 Hardware Data paths

Decrypted data shall not be present on any user accessible buses in useable form. For this purpose user accessible bus is defined as: (1) an internal analogue connector which is designed for end user upgrades or readily facilitates end user access (2) a data bus that is designed for end user upgrades or access, eg: PCMCIA, CardBus, PCI bus.

A user accessible bus does not include memory buses, CPU buses or similar portions of a device's internal architecture.

9.4 Required Levels of Robustness

The levels of robustness described in sections throughout this section shall be implemented so that it is reasonably certain that, without physically disassembling the device, they cannot be defeated or circumvented using "Widely Available Tools" or "Specialized Tools", and that only with great difficulty they can be circumvented with "Professional Tools."

9.4.1 Widely Available Tools

Widely Available Tools is defined as general-purpose tools or equipment that are widely available at a reasonable price, such as screwdrivers, jumpers, clips, file editors, and soldering irons.

9.4.2 Specialized Tools

Specialized Tools is defined as specialized electronic tools that are widely available at a reasonable price, such as memory readers and writers, debuggers, decompilers, or similar software development products.

9.4.3 Professional Tools

Professional Tools is defined as professional tools or equipment, such as logic analyzers, chip disassembly systems, or in circuit emulators.

Deleted: 2/9/2004
Deleted: 2/4/2004
Deleted: 1/23/2004